Event-based Processing of Vision Applied to Stereo Vision and Behavioural Tracking

---

Dissertation

zur

Erlangung der naturwissenschaftlichen Doktorwürde (Dr. sc. nat.)

vorgelegt der

Mathematisch-naturwissenschaftlichen Fakultät

der

Universität Zürich

von

Paul Rogister

aus

Frankreich

Promotionskomitee

Dr. Daniel Kiper

Prof. Richard Hahnloser

Prof. Tobi Delbrück

Prof. Ryad Benosman

Zürich 2010

Acknowledgements

This thesis was a great opportunity to meet and interact with many interesting researchers and students, especially as it involved many collaborations from neuromorphic engineering to behavioural experiments. I would like to thank here the many people who were instrumental in this thesis achievement and made it a wonderful experience.

Foremost I would like to thank Daniel Kiper for his kind and steady supervision. It was a few long years but always feeling his trust made things lot easier, along with his efficient comments. I am also most grateful for Ryad Benosman's help and friendship, which allowed me to pass through difficult times of doubt, and whose expertise helped me greatly improve my methods. I am glad he didn't lose that eye the day we set up a new stereo rig at the capo caccia workshop series.

This thesis was only possible thanks to Tobi Delbrück's own research and implementation of a silicon retina. His further support and enthusiasm was another essential building block of this thesis. Patrick Lichsteiner spent so much time with me to set up the experiments, build the first stereo rig especially for me, and constantly improving the hardware according to my needs, that I cannot thank him enough. Raphael Berner took up the challenge when Patrick left, and was always available for advices and support whevener needed. Kynan Eng also acted for me as an additional supervisor, always concerned about the way the thesis was going and giving very good advices and always available for talks rehearsal. It was really great to be surrounded by so many helpful and kind people.

This thesis was strongly motivated by the need to monitor rehabilitation studies for which I had no previous experience. Thanks to Martin Schwab, I was very lucky to collaborate with Irin Maier and Michelle Starkey who gave so much time, patience and attention to my work, with the very kind help of Miriam Gullo and Christiane Bleul. It was a very rich experience for me and I hope this thesis outcome can give them something back. I also followed some of Lukas Bachmann's

experiments, his intelligent setup design and sharp comments were really very useful.

All members of INI were of great support, I had a great time in this wonderful environment and to name but a few I'd like to especially thank Sergi Bermudez i Badia, Markus Baumann, Edith Chevrier, Prasanth D'Souza, Marie-Christine Fluet, Alexis Guanella, Cyrille Girardin, Vasco Medici, Srinjoy Mitra, Rico Moeckel, Henning Proske, Pawel Pyk, Milanka Ringwald, Andreas Steimer, Claude Z.H. Wang, Yingxue Wang and Frederick Zubler. I would like to thank also Rodney Douglas, Richard Hahnloser and Kevan Martin for the creation of this successful research environment, as well as Simone Schumacher, Kathrin Aguilar Ruiz-Hofacker, Stephan Rickauer and all the INI staff. This thesis was also the occasion of meeting and sharing ideas with many great people, and for this I would like to thank especially Giacomo Indiveri. I had a very good time working and discussing with Christoph Posch and Sio-Hoi Choux Ieng especially. Let me also thank my family and Virginie Boreux who were always present and essential.

This work was conducted as part of a NCCR Neural Plasticity and Repair project funded by the Swiss National Science Foundation. I want to thank Robert Riener who lead our workgroups and organized everything so neatly.

Finally I would like to thank again my committee members for their patience and interest in my work, Daniel Kiper, Richard Hahnloser Tobi Delbrück and Ryad Benosman, as well as the future readers of this thesis.

Abstract

In computer vision, the visual data is represented and manipulated as frames. Recent advances in neuromorphic engineering made it possible to process vision through a different approach, coding the data as a stream of events representing changes in brightness. The event carries information about the pixel sensing a change, but also the exact timing of this change. This event data representation is powerful, not only encoding the data in a compressed form but also providing new computational possibilities. In this thesis we study how to process this event-based data to solve computer vision tasks. We use in our experiment the dynamic vision sensors developed by Tobi Delbrück et al., providing us with visual event-based data. We first focus on adapting existing computer vision solutions dealing with spatial problems. We derive principles of design for event-based spatial convolutions and show that spatial vision tasks can be solved using event-based methods. We then address a more challenging task, stereo vision. We show how to calibrate a stereo rig of dynamic vision sensors by adapting frame-based calibration methods. The additional temporal dynamic encoded in the stream of events is exploited to solve the stereo correspondence problem in a novel way. Based on these event-based vision framework we built a prototype behavioural monitoring system applied to the rat pellet-reaching task, a model for rehabilitation studies that currently lacks accurate quantification techniques. The performances of our event-based tracking system are promising and the potential for further development shows that event-based computation can be used in challenging vision applications.

Zusammenfassung

In Computer Vision (Maschinelles Sehen) werden visuelle Daten als Frames dargestellt und manipuliert. Neuste Fortschritte im Neuromorphic Engineering machten es möglich das Sehen mit einem anderen Ansatz zu bearbeiten. Dabei werden die Daten als Ereignisflüsse kodiert, welche als Änderungen in der Helligkeit dargestellt werden. Die Ereignisse beinhalten Information über die Pixeländerung, aber auch über das genaue Timing dieser Änderung. Diese Darstellung als Ereignisdaten ist sehr leistungsstark, weil die Daten komprimiert kodiert werden und zudem neue rechnerische Möglichkeiten ermöglichen. In der vorliegenden Doktorarbeit haben wir untersucht, wie man diese ereignisbasierenden Daten bearbeitet um Computer Vision Aufgaben zu lösen. In unseren Experimenten wurde der dynamischen Vision Sensor benutzt, welche von Tobi Delbrück et al. entwickelt wurden und uns die visuellen ereignisbasierenden Daten liefert. Als erstes konzentrieren wir uns darauf bestehende Computer Vision Lösungen anzupassen welche sich mit räumlichen Probleme befassen. Wir verändern Gesetzmässigkeiten von Designs für ereignisbasierende räumliche Biegungen und zeigen, dass Aufgaben des räumliche Sehens durch ereignisbasierende Methoden bewältigt werden können. Weiterhin sprechen wir ein noch herausforderndere Thema an, Stereo Sicht. Wir zeigen auf wie man eine Stereoanlage mit dynamischen Sichtsensor kalibrieren kann indem man rahmenbasierende Kalibrierungsmethoden anpasst. Die zusätzliche temporale Dynamik welche in dem Lauf der Ereignisse kodiert ist wird ausgenutzt um Probleme mit Stereokorrespondenzen neuartig zu lösen. Basierend auf dieser ereignisbasierenden Sichtstruktur bauen wir einen Prototyp für verhaltensüberwachende Systeme welches bei Ratten Pellet-Erreichen Aufgaben angewandt wird, eine Vorlage für Rehabilitationsstudien welchen es Momentan an akuraten Quantifikationstechniken fehlt. Die Effizienz unserer ereignisbasierenden Verfolgungssystem ist vielversprechend und das Potential einer weiteren Entwicklung demonstriert dass ereignisbasierende Berechnungen zum Gebrauch in herrausfordernden visuellen Bereichen Anwendung finden kann.

Table of Contents

1

Introduction

The aim of this thesis is to investigate event-based vision computation. Event-based vision is the coding of visual signals using the event data representation. The event data representation is inspired from the neuronal spikes observed in biology. The data is encoded in sequences of spikes or "'events"' instead of passing arrays of real values. We study whether this data representation allows us to address problems too challenging for classical computing techniques, specifically here in the field of visual processing.
Some tasks such as visual recognition of objects are not yet completely solved by computers however how much time and computational power is possibly available. Some of these tasks however seem trivially performed by biological nervous systems. This lies both in hardware and software differences, which in the context of the biological nervous system refers to the neuronal substrate and to the way the data is represented and manipulated by this substrate.
My thesis focuses on the development of the event-based computation exploiting the event data representation. The derived event-based algorithms rely strongly on the progress made in the field of neuromorphic engineering that study how to physically emulate the computational principles of the nervous system. Using these principles to solve hard problems is a new and promising approach which presents also many challenges. We implement our event-based vision solutions to process the data provided by recently developed neuromorphic devices.

Our research on event-based vision computation is applied to a concrete example, the monitoring of the rat pellet-reaching task used for neuro-rehabilitation studies for which no satisfactory automated monitoring solution exists. We use a neuromorphic vision sensor that allow us to represent the problem in a code similar to the retinal code.
This chapter will introduce neuromorphic engineering, vision and the neuromorphic approach to vision sensors that provides data as a dynamic stream of events.

1.1 Neuromorphic Engineering

1.1.1 Origins

Neuroscience has lead to the identification and description of many of the nervous systems components, from the morphology of neurons and supporting cells to their genetic, chemical and electrical processes, and of their connectivity into large networks. In parallel, electronics has attempted to improve the transistor design. Despite the influence of Von Neurmann in relating biological neurons and computers [von Neumann, 1958], the development in electronics created computational devices very different from the nervous system. The increase in complexity of electronic devices and the need to integrate more components in the same system or chip led to development of more efficient integration techniques. In the 1970's the invention of the Very-large-scale integration (VLSI) technique was a improvement leading to the possibility to build very large chip like nowadays computer's processors. These electronic components have some severe limitations: their reliance on perfect signal and synchrony made them costly and inefficient to scale up [Indiveri et al., 2008][Sarpeshkar, 1998]. This increase in design complexity and the parallel advances in neuroscience led researchers to emulate more realistically the way the nervous system processes information, which is robust to noisy data signals and composed of networks of very high number of units, with around a hundred billion neurons in the human brain [Williams and Herrup, 1988]. This specific approach of using Very Large Scale Integration (VLSI) techniques to reproduce biological components was named Neuromorphic Engineering by Carver Mead in the 90s [Mead, 1990]. Unlike classical electronic engineering, neuromorphic engineering attempts to take advantage of the similarities between silicon electronic devices and biological neurons [Mead and Ismail, 1989]. These similarities span from the electric current properties of both substrates to the noise and variability inherent in both systems. Sharing the need to be adaptive, the VLSI components are thus facing the same learning challenges as biological components [Mead, 1990].

1.1.2 Analog Very Large Scale Integration

A very important aspect of the nervous system is its ability to perform not only digital computations but also analog ones. This led to the development of analog VLSI (aVLSI) incorporating transistors exploited in a sub-threshold regime that reproduces the electric properties of the neurons membranes [Liu et al., 2002].

1.1.3 Neuromorphic neurons

In the last two decades its was thus attempted to implement the nervous system's components from neurons to neuronal networks. Neurons of neuromorphic design include nowadays a wide range of models, from the simplified integrate-and-fire spiking neurons to more realistic conductance-based bursting neurons (see [van Schaik, 2001], [Wijekoon and Dudek, 2008].and [Indiveri et al., 2008], [Asai, 2004] for reviews). The more realistic models require the emulation of all computational components of the neurons, like synapses and dendrites. The synapses are the input/outputs of the neurons. They can be chemical or electric like in the first stages of the retinal processing, and connect neurons together. The synapses are one of the main vectors of plasticity in a neuronal network, defining the strength of the connection. In biology, learning and adaptation are shown to occur at synaptic location [Bliss and Lomo, 1973][Cooke and Bliss, 2006]. Short-term plasticity is the modulation of synaptic strength that the input spikes induces, whereas long-term plasticity arises from the correlation between pre- and post-synaptic activity history. Silicon synapses exist in many kinds to represent this process from simple to highly detailed models [Fusi et al., 2000], [Rasche and Hahnloser, 2001], [Kanazawa et al., 2004]. Wang et al. proposed an architecture able to control the local strength of a large number of synapses in [Wang and Liu, 2006]. With independent synapses to connect neurons together, it becomes possible to emulate plastic networks of neuromorphic neurons and study their learning properties.

1.1.4 Neuromorphic networks

Mitra and Indiveri presented [Mitra et al., 2009] a VLSI implementation of a network of integrate and fire neurons which exploits the spike-based synaptic plasticity of the network to make it learn and classify binary patterns. With Fusi [Mitra and Indiveri, 2009], they also addressed the classification of complex patterns in real-time. The study of neuronal computations such as winner-take-all or classification performed at network level with real-time speed (in opposition to the slow offline processing of software simulations) and very similar physical constraints.
The difficulties faced by the engineer were also beneficial where similar constraints affect biological neurons. One example is the challenge of computing with many but mismatched components. Like real neurons which are all different, the implementation of silicon neuron in aVLSI is prone to mismatch.. The aVLSI analog transistor, despite its accurate design, behaves with some variability [Pavasović et al., 1994]. Liu and Douglas showed in [Liu and Douglas, 2004] that despite this variability inherent to the silicon design of neurons they could

reproduce spike synchronization at the network level.
To support communication between aVLSI spiking devices, Boahen et al have modernised the address-event representation (AER) that allows the transmission of data between devices [Boahen, 1998]. The data is encoded as events described by the time of occurrence and the address of the emitting silicon neuron. Thus the connectivity can be implemented by having target neurons listen to all broadcasted address-events and filtering out those they are not connected to. This framework allows the effective scaling up of simulations by connecting many aVLSI devices together [Serrano-Gotarredona et al., 2006]. The neuromorphic engineering field is only starting to address its research possibilities, as Indiveri, Chicca and Douglas evoke in a comprehensive review [Indiveri et al., 2008]. The future developments are aimed toward the study of sensory modality and higher cognitive functions using neuromorphic implementations. Many groups focus on developing neuromorphic vision systems based on biological vision, as it is both an extensively studied field in neuroscience, and a domain in computer science where there is a high demand for more efficient applications and devices.

1.2 Vision

Vision is the process of sensing photons as a mean of revealing information about the environment. Surely the most important sensory modality for humans, it allows animals with photo-sensitive sensors to perceive distant features of the environment based on their different reflective or light-emitting properties.
We discuss here the biological retina and the field of computer vision. We will also address the fundamental difference between biological vision and computer vision, i.e. the way information is represented, by discussing the spike codes used in the brain.

1.2.1 The biological retina

In the course of biological evolution, due to the survival advantage that visual sensors can provide, different and complex techniques have appeared in the composition of these photo-sensors or eyes [Parker, 2009]. The invertebrates sport ten different eye systems [Cronin and Porter, 2009] while the vertebrates all share one similar design. This vertebrate eye design make use of a lens to focus the rays of light onto specialized layers of cells: the retina. The relation between these incoming rays of light and the properties of the external physical world is well described by optics and projective geometry [Faugeras, 1995]. In term of biophysical mechanisms, the focused ray of light reaches the photo-sensitive

cells that can react to very low light intensities, generating an electrical signal by photo-transduction. There are two types of photo-receptors, cones and rods, see figure 1.1.

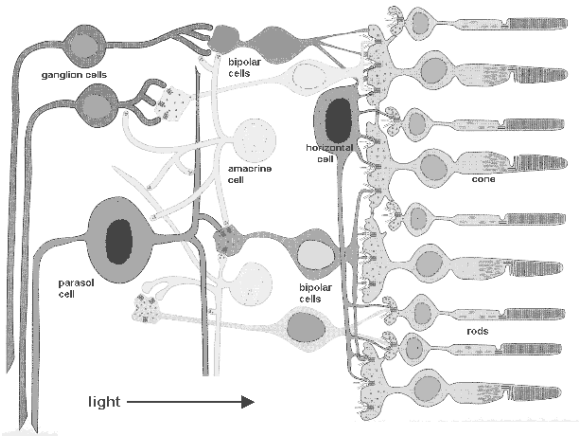

Figure 1.1: The layer of retinal cells, from [Dowling, 1997]

Primates cones photoreceptors exist in three types, long, middle and short and are sensitive to wavelength. The rods are more sensitive to low light intensity, up to a thousand time more sensitive than cones. They have a different spatial distribution in the retina, the cones being denser in the fovea and the rods more numerous in the periphery, but all are part of the retinal neural network consisting of horizontal cells, bipolar cells, amacrine cells and ganglion cells. The ganglion cells are the only one to produce spikes, the short digital impulse that can be transmitted through long axonal connections. All the communication between retinal cells are made through electrical synapses, like in analog circuits. The ganglion cells spike output is projected through the optical nerve toward the lateral geniculate nucleus of the thalamus. The neuronal signal representing the visual input is then transferred and transformed in the visual cortex, where it is processed for cognitive purposes [Braddick, 2001]. This retinal neural network is not a passive photon sensor, it pre-processes the visual information before sending it to the visual areas of the brain. This processing includes adaptation to a large range of light condition, maintaining contrast sensitivity across 12 order of magnitude in varying light intensities. The retina also heavily compresses its output: in the human retina there are 127 million input photoreceptors but only 1.25 million output ganglion cells. The ganglion cells can be further distinguished into classes by their projection target and their morphology but all ganglion cells classes share the separation into ON and OFF ganglion cells. The ON ganglion cells convey information about positive change in local luminance while the OFF ganglion cells transmit information about decrease in luminance.

These ON and OFF signals have both spatial and temporal properties. The spatial response of the ganglion cells is defined by their spatial receptive field applied to their sampling of a hundred input photoreceptors each in average, but this number varies amongst ganglion cells classes. The spatial receptive fields are often center-surround filter of various size. The temporal response of the ganglion cells is of two distinct types : the transient and the sustained responses (see figure 1.2). The sustained response is a steady output signal correlated to the stimuli length while the transient response is a short signal of the onset of the luminance change.

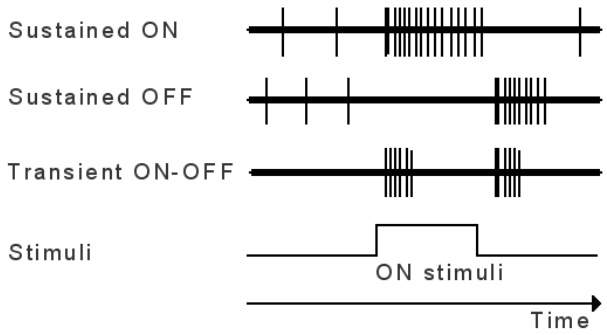

Figure 1.2: Sustained ON and OFF response and transient ON response of a ganglion cell to a positive luminance change stimulus

1.2.2 Computer vision

Far before any attempts to technologically reproduce a vision sensor had been made, the reproduction of visual perpception took the form of static pictures such as paintings. With the progress of technology, especially optics and photosensitives films, the first pictures obtained from artificial sensors were also static. Dynamic vision could then be reproduced as a succession of static images, first implemented by Joseph Plateau in 1831 [Plateau, 1839], or frames, in a movie-like sequence, like in the attempts of Louis Le Prince in 1888 [Scott, 1923] or of Thomas Edison and his kinetoscope also in 1888 [Dickson et al., 1895].The development of a camera was about synchronizing shutter speed and rotating films to produce serial impressions of the incoming rays of light. The switch from film to digital camera brought a totally different technology but it was used to perform the same exact process where captured light was summed and arrayed into frames to produce video. While these video can be recorded at very high speed, i.e. frame-rate, those are nonetheless sequences of static frames.
Computer vision is the field studying how to process and analyze images with a

computer, originally in emulation of the human ability to extract rich information from visual data, such as object recognition. But the visual input used was always an image or a video sequence of images. The reasons are that up to recently it was not possible to provide a digital visual input other than from digitized images or directly captured by digital cameras. The drive to keep experimentation simple also played a role, as psycho-physics have shown than humans could perform many visual tasks even on schematic drawings, so the first steps were logically to begin with simple static images. The processing of movement data required the use of video stream, but again these video were obtained using the inherited method of sequencing static images into succeeding frames. This implied a frame-based data representation that seemed inevitable as compression of the recorded photon stream was mandatory. Before another form of input became available, computer vision only focused on processing this specific data representation.

1.2.3 Spike data representation

The retinal output is the basis for all further computation on visual input in the brain and is represented in spikes traveling through the optic nerve. Understanding how the information is coded and compressed by the retina would support developing more efficient vision algorithms. The spike is generated near the basis of the neuron's axon, due to a change in voltage potential of the neuron's membrane that would exceed a certain spiking threshold. The spike results thus from the integration of the inputs of the neuron, as defined by the activity of its incoming synapses which are connected to previous neurons axons or to sensory modalities specific cells. As a result, a receptive field of a neuron can be characterized by the modulation of stimulation affecting its spiking output. This is more easily observed in neurons whose activity is strongly correlated to a sensory modality, like in vision where the receptive field of a neuron can correspond to location in visual space, or in auditory modality where a receptive field can be a frequency range. The existence of preferred stimulus in these receptive field leads further into the understanding of what might the neuron's spikes code for. In the visual system, orientation selective cells have been described that emits spikes only if an edge of preferred orientation is presented in their visual receptive field. By studying this neuronal codes starting in the periphery, physiological experiments revealed the first stages of functional building blocks of visual and auditory systems. The progress toward higher cognitive functions is impeded by the complexity of the connectivity and the growing distance to explicit sensory stimuli. The same applied for the study of motor output and behavior.
Without being able to directly match the neurons output with an external cue,

we can use computational models to study the possible coding schemes. The response of neurons is not limited to one spike, but form a sequence of spike discharges, defined as a spike train. [Kenyon, 2009] divides the many possible coding schemes into two classes: population code and spike correlation code. The population code is a family of coding schemes whose principles are to encode the data into the activity of a large number of neurons. The output of only one neuron would not be significant but the overall activity of a population could be sampled and carry a computational meaning. This code is very robust to noise and deficiency. Winner-take-all networks, that can retrieve the response of single units can be used to read-out this code and have been successfully implemented following biologically plausible principles. The fact that the spike response that is specific to a neuron is lost in the midst of the population response favors the hypothesis of encoding information in the rate of spike trains.
The rate-coding is a measure of the frequency of the spike train, averaging the number of spikes over time. This measure is the easiest to obtain but the need for a time window limits its speed. In the brain, neuronal responses vary in frequency from below 1Hz to above 100Hz, so depending on the read-out sensitivity to frequency differences, this code does not allow a very wide range of possible representations. It is nevertheless probably used in the brain in specific situation, like muscular control or the midbrain's activation of motor nuclei. Two spike trains of identical frequency can exhibit totally different temporal patterns. It could be that the exact patterns have a significance. In the simplest case the sequence of spikes could implement a binary code.

The other family of coding schemes studies the correlation between spike trains as a mean to convey complex data representations. The distance between spike trains can simply be measured as the synchrony between mutual spikes, if the spike train can be aligned in time. Another hypothesis is to use pattern recognition to decode the spike trains which would be categorized as distinct temporal patterns by additional neural networks (Hopfield neural networks, recurrent network attractors). When assuming that individual neurons have a specific role inside a population of neurons, the interpretation of this population's spiking train amounts to the description of spatial excitation patterns, that when correlating with each other can engage further cognitive processing.
Exact spike timing can also carry information, like in the spatial localization of the source in auditory processes using the temporal difference between the signal arrival to the two ears. This representation of the data into temporal pattern and not only spatial patterns can be read out by biologically plausible neural networks. For example the neural timing net proposed by Cariani et at [Cariani, 2001] are able to perform temporal computations on spike trains to detect and characterize their temporal patterns. This allows their model of neural network to discrimi-

nate auditory objects based on their frequency invariance. Simon Thorpe and his team focused on the importance of individual spike timing and the order of spike firing in a population to design fast classification network [Thorpe et al., 2001]. This coding scheme called Rank order code is optimized for speed and is a good candidate for early vision data representation in the context of fast identification of features [Van Rullen and Thorpe, 2001]. Amongst all the possible coding schemes based on spike trains, more than one is likely to be used in the brain as the different modalities (vision, audition,olfaction,..) likely requires different optimized schemes. The many constraints at play in biological nervous systems weigh heavily on the description of the best solution. In order to study or design efficient coding schemes to solve cognitive problems similar to those faced by the brain, it is mandatory to model these constraints as realistically as possible and to extract the principles that govern the choice of the biological solutions.
In the context of visual processing, this is the approach of Carver Mead
[Mead and Mahowald, 1988] who studied the retina by emulating its elements into silicon hardware. Neuromorphic vision sensors have been developed that provide us with devices representing visual data as spikes.

1.3 Neuromorphic Vision

1.3.1 Neuromorphic photoreceptor

Many neuromorphic implementations of vision sensors have been designed to emulate different level of visual processing, from the photo-receptor behaviour to specialized visual processing [Indiveri and Douglas, 2000]. The first neuromorphic vision sensor, implementing the first step of biological vision, was demonstrated by Mahowald and Mead in 1988 [Mead and Mahowald, 1988]
[Mahowald, 1992]. It introduced an analog photoreceptor that transforms the perceived light intensity into an output voltage following a logarithmic mapping. In classical vision sensors, the pixel response encodes the received intensity light, giving a measure of the absolute intensity value, expressed in grey levels at the moment of the non-continuous read out. For example, in some asynchronous neuromorphic designs, the pixel generates a spike as a threshold function of the relative change of luminance since the last spike, encoding the temporal contrast of the visual scene. The logarithmic transfer function effectively compresses the input and gives the sensor a large dynamic range of exploitation. The mismatch in the components led to variable responses limiting the sensitivity of the device, and the response was slow in low light conditions.
Delbrück and Mead solved these problems by adding active adaptation

[Delbruck and Mead, 1995]. They implemented an active feedback loop predicting the input signal and correcting for mismatch as well as speeding up the response. Unlike in the biological retina, where two types of photo-receptors (rods, cones) are needed to provide the necessary sensitivity range, one type of silicon pixel is enough because of their high dynamic range
[Etienne-Cummings and Spiegel, 1996]. Jorg Kramer improved the design by adding a rectification to the perceived intensity change [Kramer, 2002], emulating the biological retina and its ON and OFF cells. This design implements the address-event representation protocol so that each pixel can be addressed individually.

1.3.2 Neuromorphic retina model

The retina is not only an array of adaptive photoreceptors, but also a neuronal network achieving local computations before sending a preprocessed output to higher visual areas of the brain. These local computation include adaptation to contrast and luminance, temporal and spatial filtering of the image to reduce noise and detect motion, orientation and basic shapes. [Ruedi et al., 2003] proposed a vision sensor made of 128 by 128 pixels that compute spatial filtering at the pixel level and outputs contrast magnitude and direction of features. The output is temporally ordered to prioritize high contrast features so that further processing can be accelerated by discarding low contrast features. The sensor has a high dynamic range of 120dB but still perform frame-based without exhibiting the temporal properties of the retina. In 1996, Liu and Boahen attempted to model both spatial and temporal filters in their implementation of an adaptive retina with center-surround receptive Field [Liu and Boahen, 1996]. Zaghloul and Boahen went further by implementing the actual biological retinal cells and connections into a neuromorphic model of the retina [Zaghloul and Boahen, 2006]. It emulates not only the first stages of the photo-transduction but also the processing of the many different retinal neurons. The objective is to design a prosthetic vision sensor compatible with the real optic nerve so that it could be used to replace biological eyes. Both temporal and spatial filtering are achieved by this asynchronous vision sensor, but the signal-to-noise ratio is still large due to variability in the components behaviour. The dynamic range is also limited by the choice of a passive pixel instead of an adaptive one. In 2006, Kameda and Yagi designed a silicon retina that emulates separately the photoreceptors and horizontal cells circuits of the biological retina [Kameda and Yagi, 2006]. They used sample/hold circuits to monitor charging pixels and recreate the transient and sustained responses of the retina, albeit using synchronous pixels. The configuration of their system is very flexible and can output frames at a rate of 200 frames

per second at a resolution of 128 by 128 pixels. The interaction between the output of the photoreceptors and horizontal cells networks allows for efficient oriented movement detection.

Focusing on emulating the temporal transient responses of the retina, Kramer designed a silicon retina model using their adaptive photoreceptor into a 128 $\times$ 128 array that enforces local adaptation. The pixels perform temporal filtering by implementing the retinal transient response [Kramer, 2002]. The pixels are asynchronous and continuously generating a stream of ON-OFF events instead of frames. A similar model is for example used as a training tool, the Physiologist's Friend Chip [Delbruck and Liu, 2004]. It allows to perfect the setup in experiments studying the effect of stimuli on transient retinal responses. Lichsteiner and Delbrück improved the model from Krammer into the Dynamic Vision Sensor (DVS) [Lichtsteiner et al., 2008]. It performs asynchronously and continuously to send contrast change events using the Address Event Representation protocol through a standard USB port, making it available for use in real applications.

Posch and colleagues have pushed further the asynchronous DVS design. Their Asynchronous Time-based Image Sensor (ATIS) [Posch et al., 2010] implements the sustained response encoding the intensity in absolute grey-levels by adding a second event generator. When a change of contrast is detected above threshold, a first event is sent as with the DVS, but an intensity measurement is also started that ends by emitting a second event. The time delay between the first and second events represents the measured intensity. Their sensor array has a resolution of 304 $\times$ 240 pixels and is promising for industrial exploitation.

1.3.3 Dynamic Vision Sensors

The Dynamic Vision Sensor (DVS) (figure 1.3) is the vision sensor we used in all our experiments. It was developed by Patrick Lichtsteiner and Tobi Delbrück [Lichtsteiner et al., 2008] at the Institute of Neuroinformatics, Zurich. The sensor is an array of 128 by 128 CMOS pixels based on the pixel design of Jorg Kramer [Kramer, 2002]. Every pixel is independent and continuously samples its perceived light intensity. When the fluctuation of intensity reaches a threshold, an event is produced with its exact timing. An event can be ON or OFF according to the polarity of the intensity change. If the change is a decrease in intensity it will be an OFF event, if it is an increase it will be an ON event, see figure 1.4. This behavior is coherent with the biological responses of the retina where two distinct channels carry ON and OFF spikes to the following brain visual areas.

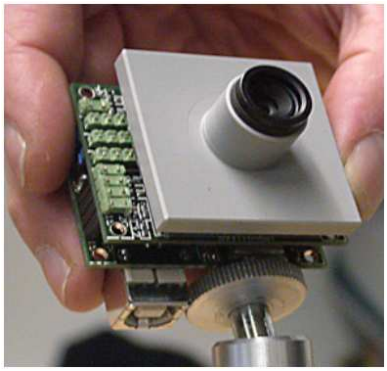

Figure 1.3: First generation DVS sensor of 128 by 128 pixels

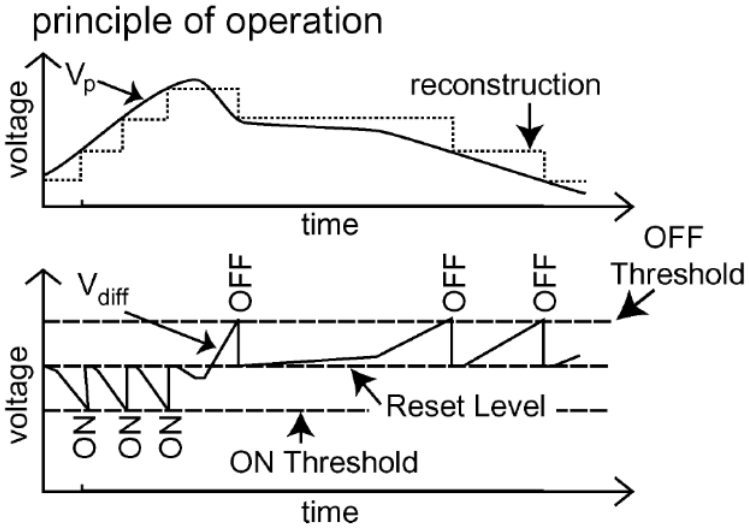

Figure 1.4: Principle of ON and OFF spikes generation of DVS pixels, from [Lichtsteiner et al., 2008]. On top, the evolution of pixel's voltage V_p. Below, the corresponding generation of ON (voltage increases above change threshold) and OFF (voltage decreases above change threshold) events, from which the evolution of V_p can be reconstructed.

The pixels are sensitive to luminance changes as small as 2 lux, to up to 100 klux, with a dynamic range of 120dB. The events are transmitted off-chip with a low latency of around 1 μs. The output is transmitted through a standard USB connection, using the Address-Event Representation (AER) protocol [Boahen, 1998], as a stream of ON and OFF events, and not as frames.
This sensor has many advantages over traditional cameras:
- only moving objects produce data

- continuous (1μs), frameless output, means high speed and continuous tracking
- low data rate (USB port) and high compression of data can record very long sequences at high speed

This vision sensor is thus particularly suited to the tracking of movement and furthermore emulates spike-based coding of the visual information.

1.4 Thesis question

State-of-the-art computer vision techniques, despite their increasing complexity, face many difficulties in solving the tracking problem. They are all based on video data obtained from digital cameras, where the input is encoded in frames. This data representation is due to technical constraints rather than effective computational power. The difficulties may actually not reside so much in the computation to execute but on the chosen data representation to manipulate in order to successfully track the target. Using neuromorphic vision sensors emulating biological systems known to outperform computer vision, it becomes possible to address the problem with a different data representation that can have computational advantages for the task. The key question of this thesis is how to process event-based data representation to improve the efficacy of computer vision tasks, and exploit both spatial and temporal information of the event-based representation.

In chapter 2 we introduce the notion of event-based algorithms. We explain how to adapt computer vision algorithms and implement spatial filters using the event-based data.

We study a more challenging task, stereoscopic vision, in chapter 3 and 4. Chapter 3 describes the calibration of event-based camera, while chapter 4 studies event-based algorithms that can take advantages of the temporal information in the event-based data to solve the stereo correspondence problem.

In Chapter 5 we apply our event-based stereo-vision algorithm to monitor an experiment using a stereo rig of Dynamic Vision Sensors.

Bibliography

[Asai, 2004] Asai, T. (2004). A neuromorphic CMOS family and its application. *International Congress Series*, 1269:173–176.

[Bliss and Lomo, 1973] Bliss, T. V. P. and Lomo, T. (1973). Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *J Physiol.*, 232(2):331–356.

[Boahen, 1998] Boahen, K. A. (1998). *Communicating neuronal ensembles between neuromorphic chips*, pages 229–259. Kluwer Academic Publishers, Norwell, MA.

[Braddick, 2001] Braddick, O. (2001). *Neural Basis of Visual Perception*, pages 16269–16274. Elsevier Ltd.

[Cariani, 2001] Cariani, P. (2001). Neural timing nets. *Neural networks*, 14(6-7):737–53.

[Cooke and Bliss, 2006] Cooke, S. F. and Bliss, T. V. P. (2006). Plasticity in the human central nervous system. *Brain*, 129(7):1659–1673.

[Cronin and Porter, 2009] Cronin, T. and Porter, M. (2009). *Visual System: Invertebrates*, pages 351–358. Academic Press, Ocford.

[Delbruck and Liu, 2004] Delbruck, T. and Liu, S. (2004). A silicon visual system as a model animal. *Vision Research*, 44(17):2083–2089.

[Delbruck and Mead, 1995] Delbruck, T. and Mead, C. (1995). *Analog VLSI phototransduction by continuous adaptive, logarithmic photoreceptor circuits*, pages 139–161. IEEE Computer Society Press, New York.

[Dickson et al., 1895] Dickson, W., Dickson, A., and Edison, T. (1895). *History of the Kinetograph, Kinetoscope and Kinetophonograph*.

[Dowling, 1997] Dowling, J. E. (1997). *Retina*, pages 571–587. Academic Press, New York, 2 edition.

[Etienne-Cummings and Spiegel, 1996] Etienne-Cummings, R. and Spiegel, J. V. D. (1996). Neuromorphic vision sensors. *Sensors and Actuators A:Physical*, 56(1-2):19–29.

[Faugeras, 1995] Faugeras, O. (1995). Stratification of three-dimensional vision: Projective, affine and metric representations. *Journal of the Optical Society of America*, 12:465–484.

[Fusi et al., 2000] Fusi, S., Annunziato, M., Badoni, D., Salamon, A., and Amit, D. (2000). Spike-driven synaptic plasticity: theory, simulation, VLSI implementation. *Neural Computation*, 12(10):2227–2258.

[Indiveri and Douglas, 2000] Indiveri, G. and Douglas, R. (2000). Robotic Vision: Neuromorphic Vision Sensors. *Science*, 288(5469):1189–1190.

[Indiveri et al., 2008] Indiveri, G., Liu, S.-C., Delbruck, T., and Douglas, R. (2008). *Neuromorphic systems*, pages 521–528. Elsevier.

[Kameda and Yagi, 2006] Kameda, S. and Yagi, T. (2006). An analog silicon retina with multichip configuration. *IEEE transactions on Neural Networks*, 17:197–210.

[Kanazawa et al., 2004] Kanazawa, Y., Asai, T., Ikebe, M., and Amemiya, Y. (2004). A novel CMOS circuit for depressing synapse and its application to contrast-invariant pattern classification and synchrony detection. *International Journal of Robotics and Automation*, 19(4):2716–2718.

[Kenyon, 2009] Kenyon, G. (2009). *Retinal Models*, pages 245–253. Elsevier Ltd.

[Kramer, 2002] Kramer, J. (2002). An ON/OFF Transient Imager with Event-Driven, Asynchronous Read-out. *IEEE International Conference on Circuit and Systems, ISCAS02*, 2:165–168.

[Lichtsteiner et al., 2008] Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128*128 120dB 15us latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid State Circuits*, 43(2):566–576.

[Liu and Boahen, 1996] Liu, S.-C. and Boahen, K. A. (1996). *Adaptive retina with center-surround receptive field*, pages 678–684. MIT Press, Cambridge MA.

[Liu and Douglas, 2004] Liu, S.-C. and Douglas, R. (2004). Spike synchronization in a network of silicon integrate-and-fire neurons. *IEEE International Symposium on Circuits and Systems*, pages 397–400.

[Liu et al., 2002] Liu, S.-C., Kramer, J., Indiveri, G., Delbruck, T., and Douglas, R. (2002). *Analog VLSI: Circuits and Principles*. The MIT Press.

[Mahowald, 1992] Mahowald, M. (1992). *VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function*. PhD thesis, Pasadena, California.

[Mead, 1990] Mead, C. (1990). Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10):1629–1636.

[Mead and Ismail, 1989] Mead, C. and Ismail, M. (1989). *Analog VLSI Implementation of Neural Systems*. Kluwer Academic Publishers, Boston, MA.

[Mead and Mahowald, 1988] Mead, C. A. and Mahowald, M. (1988). A silicon model of early visual processing. *Neural Networks*, 1(1):91–97.

[Mitra et al., 2009] Mitra, S., Fusi, S., and Indiveri, G. (2009). Real-Time Classification of Complex Patterns Using Spike-Based Learning in Neuromorphic VLSI. *IEEE Transactions on Biomedical Circuits and Systems*, 3(1):32–42.

[Mitra and Indiveri, 2009] Mitra, S. and Indiveri, G. (2009). Spike-based synaptic plasticity and classification on VLSI. *Neuromorphic Engineer*, 10.2417(1200904.1636).

[Parker, 2009] Parker, A. R. (2009). On the origin of optics, Article in Press. *Optics & Laser Technology*.

[Pavasović et al., 1994] Pavasović, A., Andreou, A. G., and Westgate, C. R. (1994). Characterization of subthreshold MOS mismatch in transistors for VLSI. *The Journal of VLSI Signal Processing*, 8(1):75–85.

[Plateau, 1839] Plateau, J. (1839). *Memoire Sur L'Irradiation*. Kessinger Publishing.

[Posch et al., 2010] Posch, C., Matolin, D., and Wohlgenannt, R. (2010). A QVGA 143dB dynamic range asynchronous address-event PWM dynamic image sensor with lossless pixel-level video compression. *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*, pages 400–401.

[Rasche and Hahnloser, 2001] Rasche, C. and Hahnloser, R. H. (2001). Silicon synaptic depression. *Biological cybernetics*, 84(1):57–62.

[Ruedi et al., 2003] Ruedi, P. F., Heim, P., Kaess, F., Grenet, E., Heitger, F., Burgi, P. Y., Gyger, S., and Nussbaum, P. (2003). 128x128 pixel 120-dB dynamic-range vision-sensor chip for image contrast and orientation extraction. *IEEE J. Solid-State Circuits*, 38(12):2325–2333.

[Sarpeshkar, 1998] Sarpeshkar, R. (1998). Analog versus digital: extrapolating from electronics to neurobiology. *Neural computation*, 10(7):1601–38.

[Scott, 1923] Scott, E. K. (1923). The Pioneer Work of Le Prince in Kinematography. *The Photographic Journal, UK*, 63:373–378.

[Serrano-Gotarredona et al., 2006] Serrano-Gotarredona, R., Oster, M., Lichtsteiner, P., Linares-barranco, A., Paz, R., Gomez-rodriguez, F., Riis, H. K., Delbruck, T., Liu, S. C., Zahnd, S., Whatley, A. M., Douglas, R., Hafliger, P.,

Jimenez-moreno, G., and Civit, A. (2006). AER building blocks for multi-layer multi-chip neuromorphic vision systems. *Advances in Neural Information Processing Systems*, 17(1):1217–1224.

[Thorpe et al., 2001] Thorpe, S., Delorme, A., and Van Rullen, R. (2001). Spike-based strategies for rapid processing. *Neural Networks*, 14(6-7):715–725.

[Van Rullen and Thorpe, 2001] Van Rullen, R. and Thorpe, S. J. (2001). Rate Coding Versus Temporal Order Coding: What the Retinal Ganglion Cells Tell the Visual Cortex. *Neural Computation*, 13(6):1255–1283.

[van Schaik, 2001] van Schaik, A. (2001). Building blocks for electronic spiking neural networks. *Neural networks*, 14(6-7):617–628.

[von Neumann, 1958] von Neumann, J. (1958). *The computer and the brain*. Yale University Press, New Haven, CT.

[Wang and Liu, 2006] Wang, Y.-X. and Liu, S.-C. (2006). Programmable synaptic weights for an aVLSI network of spiking neurons. *IEEE International Symposium on Circuits and Systems*, pages 4531–4534.

[Wijekoon and Dudek, 2008] Wijekoon, J. H. B. and Dudek, P. (2008). Compact silicon neuron circuit with spiking and bursting behaviour. *Neural networks*, 21(2-3):524–534.

[Williams and Herrup, 1988] Williams, R. W. and Herrup, K. (1988). The control of neuron number. *Annual review of neuroscience*, 11:423–53.

[Zaghloul and Boahen, 2006] Zaghloul, K. a. and Boahen, K. (2006). A silicon retina that reproduces signals in the optic nerve. *Journal of neural engineering*, 3(4):257–67.

2

Event-Based Algorithms

2.1 Spike based vision and its exploitation

Using the event data representation instead of frames changes the way visual computation is done. All previously described computer vision solutions are designed for frame-based data. In this chapter we are interested in showing if and how it is possible to transfer computer vision methods to event data representation. We believe the event data representation is a more general and powerful representation than the frame-based representation, this would lead to the possibility to re-use existing solutions and to derive new ones. Moreover, the neuromorphic implementation of biophysical principles into spiking silicon neurons gives us the possibility to study the efficiency of different spike-based coding schemes not only through theoretical models but in developing concrete applications and solutions.
We focus here thereby on the design of computer vision algorithms manipulating this spike-based data, implementing them to process data from available neuromorphic cameras. We will use the dynamic vision sensors (DVS) in this chapter's experiments.

2.1.1 Processing the visual input

Usually an image I is defined by a frame of pixel values evaluated at time t as in equation 2.1

$$I(t) = \{f(x, y, t)\} \text{ where } x \in N \text{ and } y \in M \tag{2.1}$$

where $f(x, y, t)$ is the perceived illuminance for pixel x, y at time t and N, M the dimension of the image.

Vision can bypass the notion of frame to instead encode the visual scene as a sequence of "'events"'. In the human eye, we can distinguish two kind of retinal

responses outputted as spike trains : the transient and the sustained responses [Parker, 2009]. The sustained responses gives a relative measure of the perceived light intensity, while the transient response encodes for local changes of luminance. In this thesis, we restrict ourselves to the study of transient event streams.

A transient visual event is a polarized event emitted when a relative change of illuminance is detected by the pixel. The value of the event is either -1 for a negative change of contrast of +1 when a positive change of illuminance is detected. The change of illuminance is computed by each each pixel comparing their current log intensity value against their previous memorized one. We denote this comparison $c(x, y, t)$ where x, y are the coordinates of a pixel and t the time at which the comparison is performed. The event ouput of the pixel is defined by equation 2.2.

$$e(x, y, t) = \delta(t, t_k).sign(c(x, y, t)) \tag{2.2}$$

where $sign$ is the sign function giving values in $[-1, 1]$ and $\delta(t, t_k)$ is the Kronecker delta function defined in its discrete version by equation 2.3. It is used to compare the time t, at which a constrast change in the scene occurs, to the time t_k at which the pixel detects the contrast change.

$$\delta(t, t_k) = \left\{ \begin{array}{l} 0 \text{ for } t \neq t_k \\ 1 \text{ for } t = t_k \end{array} \right\} \tag{2.3}$$

Thus an event represents a change of luminance in the scene, due to movement or change in the light's conditions since the last pixel read-out. The asynchronous processing of the camera allows to transmit these events in a sequence rather than in a frame, thus the exact timing of each contrast change is preserved and can be used for computational purpose. The image I is replaced by a stream S of events as defined in equation 2.4

$$S(t) = \{e(x, y, t)\} \text{ where } x \in N \text{ and } y \in M \text{ and } t \in T \tag{2.4}$$

where T is the ensemble of time $\{t\}$ during which the sensor is active. The event as defined by equation 2.2 can take the values -1, 0 and +1 for respectively negative contrast change, absence of event and positive contrast change. The absence of event when no change of contrast is detected implies that redundant visual information usually recorded in frames is not carried in the stream S of events. Thus the data compression of the event-based representation over frame-based data is improved.
The stream of events is a fully spatio-temporal representation of the data that can be manipulated as such. Figure 2.1 shows the spatio temporal visualization of the event-based visual data.

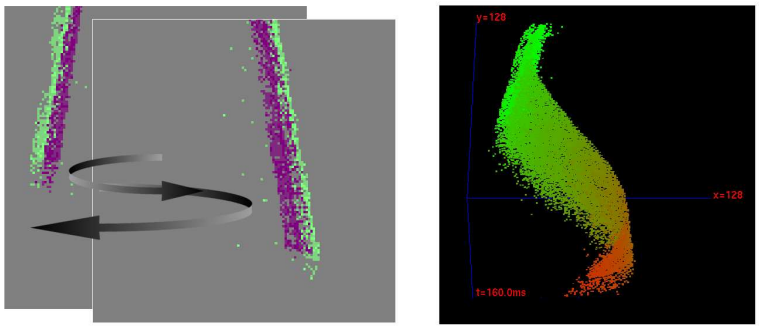

Figure 2.1: a) incoming transient events raw data plotted in 2D, a pen balancing sideways in a plane, as indicated by the arrow, b) event data representation of the movement in spatio temporal referential (x,y,t) over 160ms

Unlike the processing of image I, the processing of the visual event stream S can exploit the additional time dimension t. In this thesis we will conduct all our experiments using the asynchronous Dynamic Vision Sensors presented in section 1.3.3 that emulates the retinal transient response.

2.1.2 Existing solutions using DVS

Tobi Delbrück et al. implemented event-based algorithms that process the stream of visual event obtained from the Dynamic Vision Sensor. Their main focus is on locally processing each incoming events. They do not adapt frame-based computer vision algorithms but implement mathematical methods compatible with the event-based paradigm. The event-based 2D mean-shift tracker is a representative example of this approach [Litzenberger et al., 2006]. It is based on the mean-shift algorithm proposed by Fukugana in 1975 [Fukunaga et al., 1975] and described by equation 2.5

$$x \mapsto m(x) = \frac{\sum_{x_i \in N(x)} K(x_i - x) x_i}{\sum_{x_i \in N(x)} K(x_i - x)} \tag{2.5}$$

where x is the point in space on which the tracker is centered, x_i is a point belonging to an object to track, $K(x_i - x)$ defines a weight for x_i, typically a Gaussian shaped distance function and where $N(x)$ defines neighboring points of x for which $K(x) \neq 0$.

The algorithm requires an iteration for $x \mapsto m(x)$. In the case of event-based data, that is replaced by continuously updating x for each incoming event falling into

$N(x)$. $N(x)$ is here defined as a 2D area which shape and size can be adaptively modified according to the statistics of the event stream input.
Industrial application have been successfully designed that exploit DVS cameras, such as developed by the Austrian Institute of Technology (AIT) through their Smart Systems division. Their applications take advantage of the high temporal resolution and dynamic range as well as the high data compression rate and deliver robust tracking solutions, see [Belbachir et al., 2007].
[Perez-Carrasco et al., 2008] described a hardware event-based convolution running on continuous stream of events. Their approach highlights the advantage of event-based computation but does not take into account the dynamics of the scene.
In the present chapter we study and show how to adapt computer vision methods using local spatial information without recreating frames. We focus especially on convolving filters making extensive use of spatial frame-based information. Section 2.2 converts a frame-based spatial low pass filter to event-based data. Section 2.3 proposes an event-based thinning algorithm.

2.2 Event-based low pass filter

2.2.1 The classic spatial low pass filter algorithm

Spatial low pass filtering is a good example of simple vision data processing involving spatial computation. Generally, low pass filtering can be applied in many different fields and consists of removing as much as possible data of a frequency above a given limit, so only the lower frequencies are retained. Spatial low-pass filters can smooth the image and reduce its noise by attenuating high frequencies, see [Pratt, 1991]. The most used spatial low pass filters are the Mean filter [Gonzalez and Woods, 2002], the Median filter [Davies, 2005] and the Gaussian filter [Lindenbaum et al., 1994]. These filters all convolve a spatial kernel to compute the filtered output based on the local neighborhood of the target pixel. We focus this section on the uniform mean filter.

2.2.2 Frame-based spatial mean filter

Mean filtering is convoluting with a spatial kernel of a particular size and shape of the neighborhood. For each pixel of the image, it modifies its value as the mean value of the neighboring pixels. The spatial mean filter is mostly used to reduce noise in the image, but is suboptimal in two cases. A noisy pixel with a high value will affect its neighbors, and the edges are blurred by the convolution of the

kernel between edge points and non-edge points. A solution to these problems is to use the spatial median filter, but it is more computationally intensive (it needs to compute the median for each convolution [Davies, 2005]). In our case we want to use the filter to remove high frequency details so that large features become more distinct, and for this purpose the mean filter is more efficient than the median filter [Gonzalez and Woods, 2002].
The equation for the spatial kernel convolution is given by equation 2.6

$$g(x,y) = \sum_i \sum_j f(x-i, j-i) * h(i,j) \tag{2.6}$$

where $*$ is the convolution function, g is the output value of pixel x,y, f the input image and i,j indexes of the kernel h, which in the case of the mean filter with a 3 by 3 kernel is defined in equation 2.7

$$h = 1/9 \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{2.7}$$

The corresponding algorithm implies looping through all pixels of the image frame:
For each pixel in the array:
- compute mean value from original neighboring pixels values as defined by the kernel centered on the current pixel
- store the mean value as the new pixel value, but keep the original value for inclusion into surrounding pixels mean computation.

2.2.3 Event-based algorithm

In the classical computer vision algorithm, the spatial low pass filter is based on the spatial properties of the data, represented by the values of neighboring pixels. In the case of event-based data, we could of course reproduce frames by accumulating incoming events over time, but this would be merely falling back to the frame representation.
The properties of the event-based data representation are particularly suited for computing vision task involving precise time dynamics, like in processing movement, but we show here that it is also adapted to spatial computation.

The event-based mean filter focuses on executing local computation at the event level. It is not based on looping through all pixels values as in a frame based

algorithms. Instead, it processes each incoming event as it arrives regardless of its spatial position.

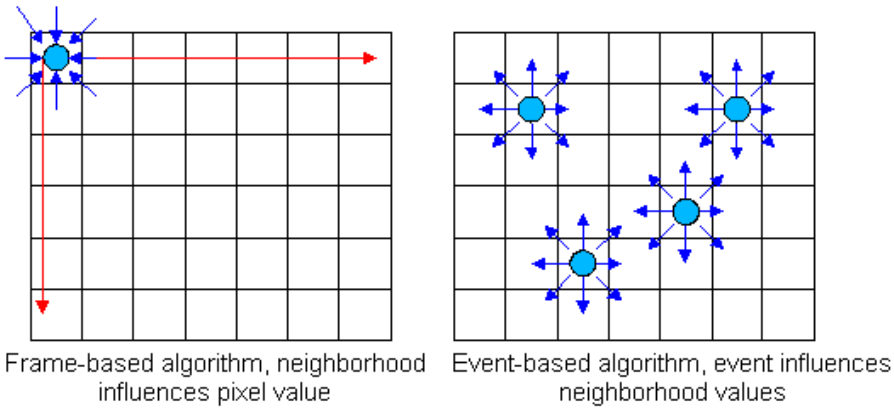

Figure 2.2: Comparison between frame-based and event-based spatial filter algorithms

We define a spatial kernel for the low pass filter, describing the neighboring relationships between pixels, in the same way as in the frame-based algorithm. Thus every incoming event affects its neighborhood as defined by the kernel, modifying an accumulated value that, with enough incoming events, will be representative of the mean filtered value of the pixels.
Equation 2.8 describes how the output value of a pixel $g(x, y)$ is affected by the occurences of events from pixel $f(i, j)$. According to the distance between pixels g and f, the value of pixel g is the sum of the weighted value of all events from f close enough to affect g. The matrix h describe these weights.

$$g(x - i, y - j) = \sum_i \sum_j f(x, y) * h(i, j) \tag{2.8}$$

The weight matrix h is the same as for the convolution kernel of classic spatial mean filter defined in equation 2.7

So the principle is reversed compared to frame-based: instead of modifying the current pixel value based on the mean value of the kernel's elements, it is the current event value that modifies the values of the surrounding pixels. Because we are dealing with events, the sum over the spatial neighborhood transforms into the sum of events occuring in this neighborhood, as shown in equation 2.9.

$$g(x - i, y - j) = \sum_t e(f(x, y)_t) * h(i, j) \tag{2.9}$$

where $e(f(x,y)_t)$ is the event occuring at time t.
The computation of g involves an additional mechanism of memory for the event values, or accumulation, so as to take into account all events occuring in its neighborhood.

Time window and decay of accumulated values

The event-based spatio-temporal convolution requires the use of a sliding time window. We implement the sliding time window by decaying accumulated values of the events over time. We accumulate values about each pixel by summing up the incoming event's polarities as defined by equation 2.10

$$v(x,y)_t = v(x,y)_{t-1} + \alpha sign(e(x,y)_t) \tag{2.10}$$

where $v(x,y)_t$ is the accumulated value at time t, $t-1$ represents the previous occurence and not the exact time value, $sign$ is the polarity of event $e(x,y,)_t$ in the range [-1 1] and α is a step value inferior to 1, chosen as $1/3$ here.
The values obtained by accumulation are not gray levels truthfully matching the intensity of the scene as recorded using a classical camera, due to the local luminance adaptation and the fact that only contrast changes are perceived.
We use a linear decay function enforcing a convergence of the accumulated value toward a reference value, here 0.5 which is the gray background value used for display. The decay function of the value v is defined by equation 2.11

$$decay(v) = v(t-1) - \lambda(t) \tag{2.11}$$

where $\lambda(t) = \alpha(t - t_{-1})/\beta$ and $t-1$ indicates the previous occurence .

$v(t)$ is the value of the accumulation at time t and $\lambda(t)$ is the linear function calculating the decay amount. α is the decay convergence constant chosen as 0.1. The decay amount is the fraction of the difference between current time t and last accumulated event time, denoted $t-1$, divided by the decay time bin β.
To enforce convergence toward the neutral value 0.5, we use equation 2.12 , inverting the sign of the operation, when the value is below 0.5. We also enforce boundaries to the operation so that the result is limited to the range [0 1].

$$decay(v) = \left\{ \begin{array}{l} v(t-1) - \lambda(t) \text{ if } v > 0.5 \\ v(t-1) + \lambda(t) \text{ if } v < 0.5 \end{array} \right\} \tag{2.12}$$

The pseudo-code algorithm for the decay function is:

```
decay( value, time ){
  timeDiff = time - previousTime
  if(value>0.5){ // converge toward reference
    res = value - (alpha * timeDiff/beta);
    if(res<0.5f) res = 0.5f;
    if(res>value) res = value;
  } else if(value<0.5){
    res = value + (alpha * timeDiff/beta);
    if(res>0.5f) res = 0.5f;
    if(res<value) res = value;
  }
  return res;
}
```

The output of the filter is a stream of generated events carrying the low pass features of the scene. This generation is done in a second loop that convolves incoming events with their pixel's accumulated low pass filtered values. For each event, if the accumulated value is of the same polarity (i.e., above medium grey level (0.5) for positive event, below medium grey level for negative events), a new event of identical polarity is generated with a probability related to the accumulated value. There is a higher chance of generating an event if the accumulated value if further away from the medium grey level value.
The pseudo-code algorithm for this filter is as follows :

```
// low pass filtering on accumulator
float accumulator[][]
for each event e
  accumulator[e.x][e.y] = (e.polarity - 0.5)*step;
  for all neighbors i,j defined by the kernel k
    accumulator[i][j] = decayed(accumulator[i][j],e.time)
    + accumulator[e.x][e.y]*k[i][j];
  end for
end for
// generation of new events
for all events
  // if polarity matches with accumulated value sign
  if accumulator[e.x][e.y]*e.polarity>0
    generate new event with random probaility depending on accumulated value;
  end if
end for
```

The implemented algorithm for the entire filter can be found in the file net.sf.jaer.eventprocessing.filter.LowPassFilterEventGenerator.java within the jAER framework.

2.2.4 Experimentation with DVS data

We implemented both frame-based and event-based low pass filters into the java framework exploiting previously recorded data from Dynamic Vision Sensors. The frame-based filter is applied after converting the DVS input into frames by accumulating the incoming events into 2D histograms.

2.2.5 Results

We present in this section the results of the event-based low pass filter on the data obtained from a DVS camera.
We use the linear decay defined by equation 2.12 to implement a sliding time window. By choosing a decay time bin value of 20ms we can process a similar amount of information as in the 20ms long frames, see figure 2.3

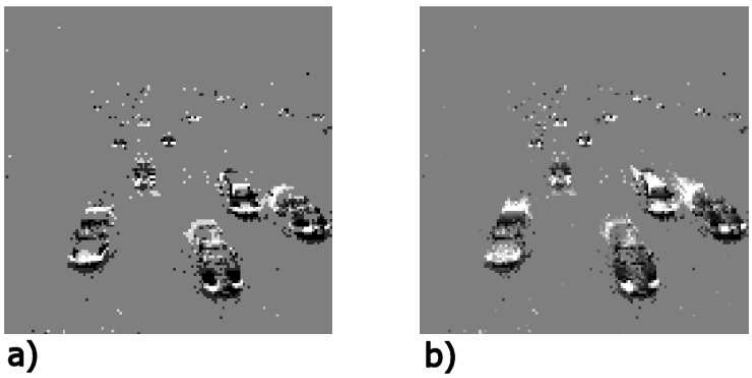

Figure 2.3: a) Cars on a highway, accumulated into one frame every 20ms, b) decayed event-stream with 20ms decay time bin

We applied the frame-based and the event-based low pass filters with a 3×3 kernel. The result is shown in figure 2.4.
We focus on the detail of one car to highlight the differences between the raw data, the effect of sub-sampling and frame-based and event-based low pass filters in figure 2.5.
We apply our event-based low-pass algorithm on a uniform scene (see figure 2.6) in order to test its effect on noise reduction. The grey values are 100%, 80%, 70%, 60%, 40%, 30%, 20%, 10% and 0% from bottom right to top left. We place an external black cardboard shutter in front of the DVS and remove it. The generated events are accumulated over 1s to capture a frame of data. The grey level obtained are only approximation of the real grey levels because of the operational principles of the DVS. The accumulation of the events over 1 second is visible in

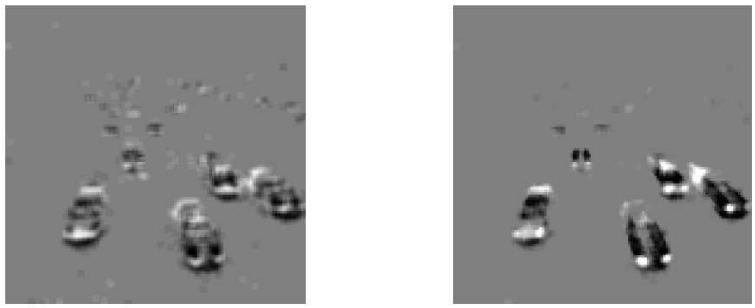

Figure 2.4: a) Frame-based low pass filtered with kernel 3×3, b) Event-based low pass filtered with kernel 3×3

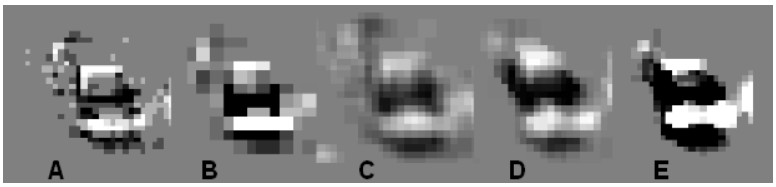

Figure 2.5: Zooming on one car, a) Raw data accumulated over 20ms, b) Data subsampled by 2, c) Frame-based low pass filtered with kernel 3×3, d) Event-based low pass filtered with kernel 3×3, e) Generated events from event-based low pass filter result

figure 2.7 a). The event low-pass filter results are shown in figure 2.7 b). The histograms of obtained intensities in each uniform region is shown in figure 2.8 for the unprocessed data, and in figure 2.9 after filtering. These histograms shows the effect of the low pass filter on removing noise and smoothing the data.

Spatial low-pass filters are evaluated on their ability to remove noise in an image. Removing noise consists of reducing the measured variation around the mean ideal intensity values. With a frame-based camera, the sampling of the image should give for each homogeneous square a gaussian distribution of measured values around the mean grey value of the square. The noise removal is efficient when the variation is reduced and thus when the width of the Gaussian diminishes. The ideal case is thus a Dirac delta positioned at the grey value. The histogram of the raw data is composed of many distinct peaks but this is not due to a low noise in the sampling. The peaks are due to the fact that the grey levels are estimated from the count of incoming events. Furthermore when looking at the histogram of a single square as in figure 2.10, we can see that not only one peak but six peaks are present, meaning that the actual Gaussian fit is rather large. Figure 2.11 is the histogram of the result of the low pass filter on this same region.

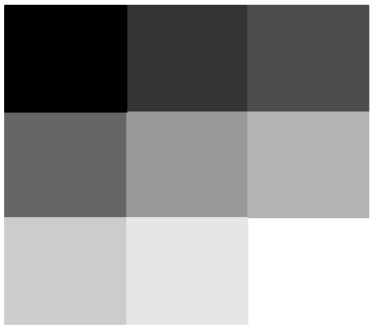

Figure 2.6: This grey scale pattern is shown in front of the DVS and recorded by removing an external shutter in front of the DVS. The shutter is a black panel and the events generated will reflect the difference of intensity between the grey scale pattern and the shutter.

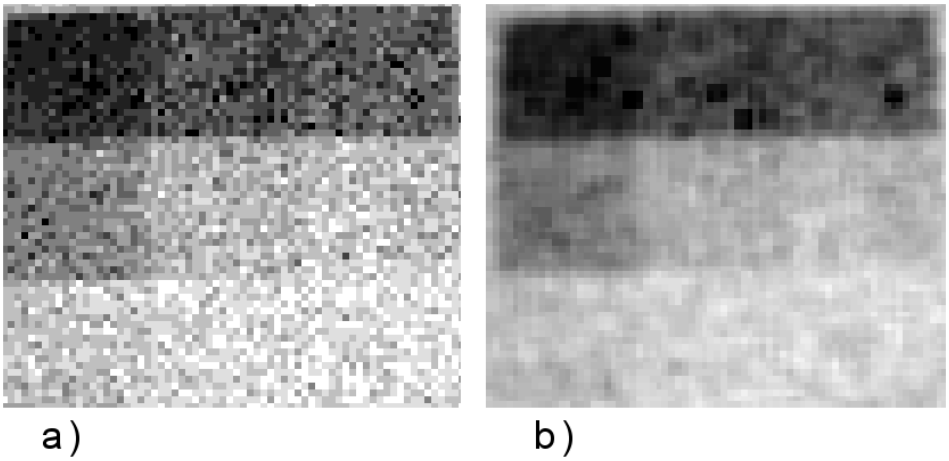

Figure 2.7: Events are accumulated over 1s after a black shutter is removed and encode the difference of intensity between the grey scale pattern and the shutter. Because the shutter is black and removed, only ON events are generated and accumulated here. Black pixels in these pictures represent the absence of ON events, while levels of grey from dark to white represent increasing numbers of ON events per pixel. a) raw data, b) event-based low pass filtered data using a 3×3 kernel.

Here the Gaussian is more apparent and actually reduced in width.

We computed for each grey value region the sigma values of the fitted Gaussian for raw data in red, and event-based low pass filtered data in blue and plotted them in figure 2.12. We can see that the filter is indeed removing noise from the data, as shown by the reduction in width of the fitting Gaussians, and further-

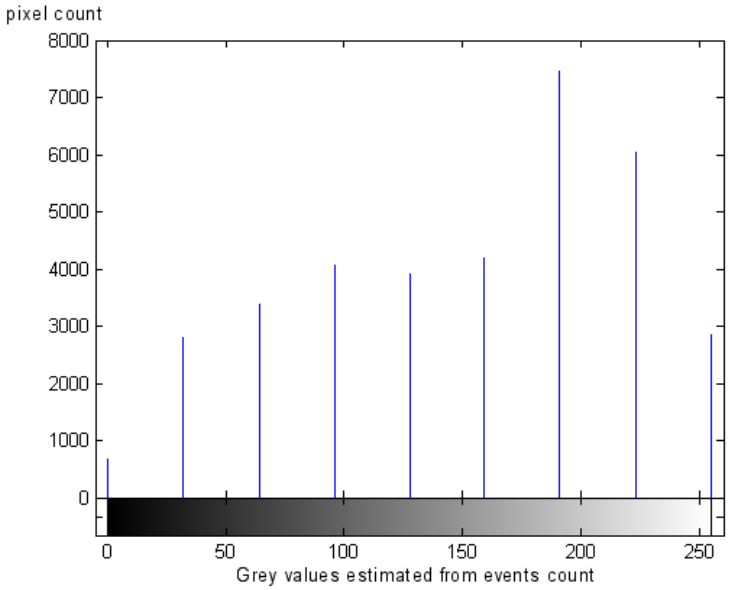

Figure 2.8: Histogram of grey level intensities present in raw capture of the grey-scale pattern. There are 256 possible grey levels in the histogram but the nine different grey regions are encoded here as a combination of nine different event counts.

more that this effect is most prominent for higher number of incoming events.

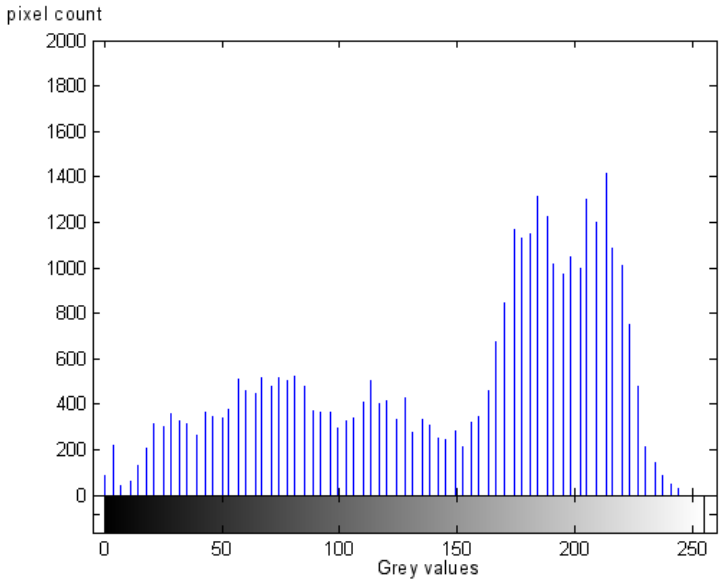

Figure 2.9: Histogram of grey level intensities present in the low pass filtered accumulation of grey-scale values. The peaks correspond to the new distribution of grey scale values in the accumulator, which has a lower resolution than the 256 possible values.

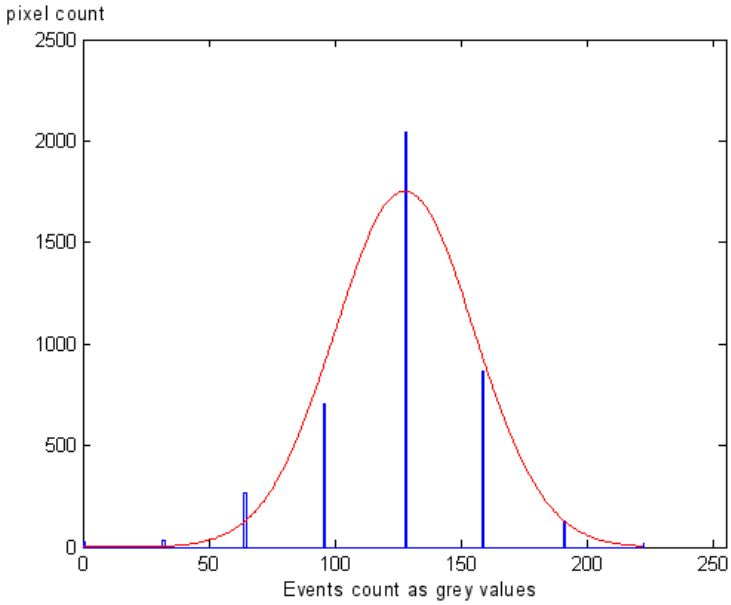

Figure 2.10: Histogram for 30% grey level, as estimated from events by accumulation over 20ms. The red curve is the gaussian fit of this histogram, indicating the overall perceived grey value.

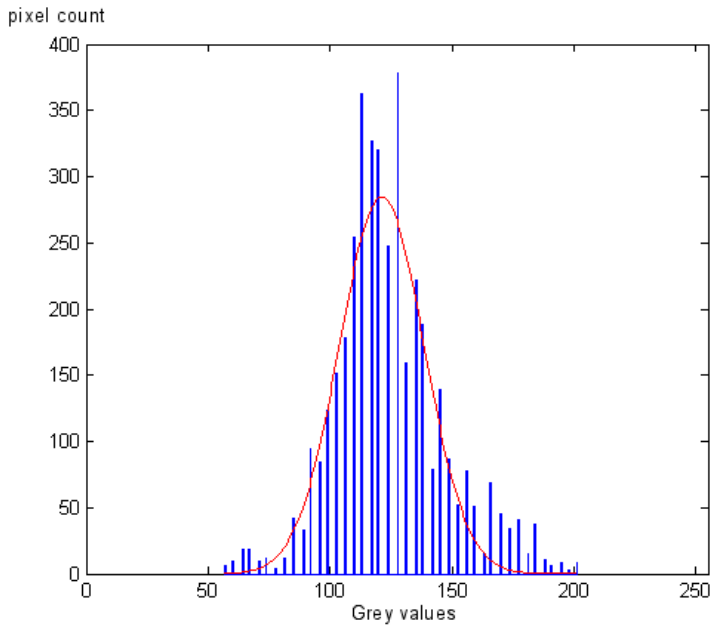

Figure 2.11: Histogram for 30% grey level, low pass filtered. The red curve is the gaussian fit of this histogram that indicates the overall perceived grey value. The gaussian fit is narrower after low pass filtering.

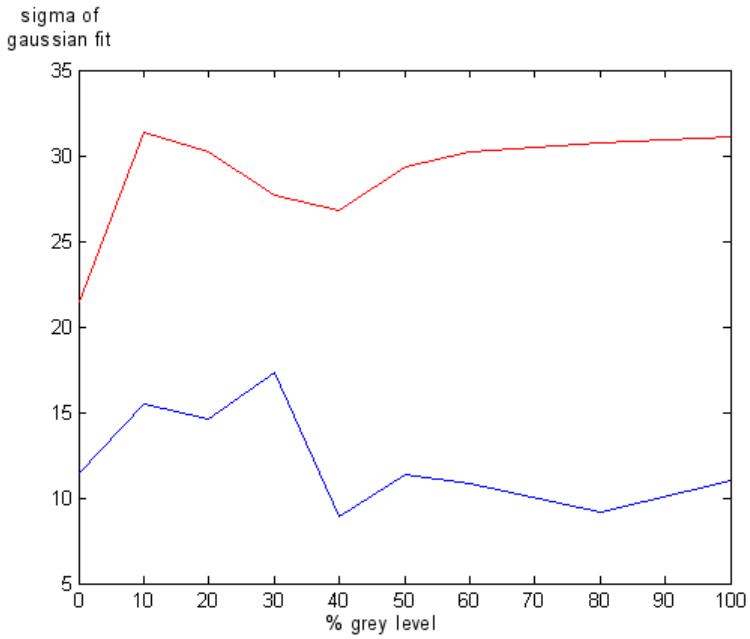

Figure 2.12: Width of the gaussian fit as a function of grey value in %. In red the raw data and in blue the event-based low pass filtered data. The reduction in width of the Gaussian fit is expected from an efficient low pass filter.

Limitations

The spatial interaction between events is defined within a time window. Fast movement within this time window will affect the result of the event-based low-pass filter. The artifacts are visible as traces behind the cars for example in figure 2.4. The validity of the time window's size depends on the speed of objects in the scene.

2.3 Event-based thinning algorithm

The thinning algorithm is another classical computer vision process. In this section we show how to adapt one of the frame-based thinning algorithms to the event data representation.

2.3.1 The thinning algorithm

Thinning is the process of reducing the thickness of the visual scene's components [Sherman, 1959] [Davies and Plummer, 1981]. When the components of the visual scene are composed of thick segments, thinning leads to the thinnest corresponding segments preserving the scene's structure. The aim is enhanced feature extraction and post-processing of the image. There exist a variety of thinning algorithms, which can be separated in two categories : those making use of local spatial information [Zhang and Wang, 1992] [Suen and Wang, 1994], and those using global information [Lin and Chen, 1995] [Melhi et al., 2001] or neural networks such as Pulse coupled neural networks (PCNN) [Gu et al., 2004] [Shang and Yi, 2007]. The challenge in the design of thinning algorithms resides in obtaining both accuracy and speed. The use of parallel algorithms has been particularly emphasized for speeding up thinning methods using local spatial information [Zhang, 1997].
We show in the next section how to convert a parallel thinning algorithm to event-based processing.

2.3.2 Frame based thinning

One efficient thinning algorithm that only relies on spatial neighborhoods was proposed in 1984 by [Zhang and Suen, 1984]. It operates on binarized images obtained by thresholding the images so that every pixel is encoded as 0 or 1. The algorithm is parallel and runs over two iterations. Both iterations perform the same computation on the neighborhood defined by 2.13 as a spiral around the centered pixel p_1. The iterations delete pixels that satisfy a set of conditions, one specific set per iteration and all conditions of the set must be satisfied. The set for the first iteration is defined by 2.14 and the second iteration's set of conditions is defined by 2.15.

$$\begin{array}{ccc} p_9(i-1,j-1) & p_2(i-1,j) & p_3(i-1,j+1) \\ p_8(i,j-1) & p_1(i,j) & p_4(i,j+1) \\ p_7(i+1,j-1) & p_6(i+1,j) & p_5(i+1,j+1) \end{array} \tag{2.13}$$

where p_k are labeled for the neighbor pixels and i, j are indexes in the 3×3 kernel centered on the current pixel being processed.

$$\begin{aligned} &\text{a) } 2 \leq BP(p_1) \leq 6 \\ &\text{b) } A(p_1) = 1 \\ &\text{c) } p_2 \times p_4 \times p_6 = 0 \\ &\text{d) } p_4 \times p_6 \times p_8 = 0 \end{aligned} \tag{2.14}$$

where p_1 is the target pixel as defined in 2.13, $BP(p_1)$ is number of non-zero neighbors of p_1, $A(p_1)$ is the number of "'01"' patterns detected in the sequence of spiraling pixels p_2 to p_9 as defined in 2.13.

$$\begin{aligned} &\text{a) } 2 \leq BP(p_1) \leq 6 \\ &\text{b) } A(p_1) = 1 \\ &\text{c) } p_2 \times p_4 \times p_8 = 0 \\ &\text{d) } p_2 \times p_6 \times p_8 = 0 \end{aligned} \tag{2.15}$$

where the only conditions $c)$ and $d)$ differ from set 2.14.

The first iteration thus only removes pixels belonging to the north-west corner and south-east boundary of the 3×3 kernel centered on the current pixel, while the second iteration deals with the remaining points. The two iterations are separated into two successive processes checking a different set of conditions to preserve the connectivity of the result. The result is an image where segments are reduced to their thinnest form. The frame-based operation of this filter involves processing each pixel of the frame in succession. This algorithm is parallel as the computation for a pixel at iteration n depends only on the its value and its neighbors values at the iteration $n - 1$. All pixels of the frame can then be computed simultaneously.

2.3.3 Event based thinning

The adaptation of the thinning filter from Zhang et al. consists in performing the computation at the local level of the incoming events. The incoming event $e(x, y)_t$ triggers two successive local computations corresponding to the two iterations of the frame-based algorithm. In the event-based version the iterations disappear to be replaced by a local two stages conditions checking for each incoming event.
The event-based algorithm uses an analog mask $B = [X, Y]$ to define the thinned components of the image. Each incoming event is associated with a value in this mask. The first stage checks if the mask point associated to the event satisfies the conditional set 2.14, defining values in B as described by equation 2.16.

$$\begin{aligned} B(x,y)_0 &= 0 \\ B(x,y)_t &= thin_1(e(x,y)_t) \end{aligned} \tag{2.16}$$

where $thin_1(e(x,y)_t) = 0$ if $N(B(x,y))_t - 1$ verifies the conditional set 2.14, 1 otherwise. $N(B(x,y))$ is the neighborhood of x, y in B, defined by 2.13.

The same check is done for all 8 neighboring elements of $B(x,y)$.

This mask is then modified by a second stage check defined by equation 2.17 that checks the conditional set 2.15.

$$B(x,y)_t = thin_2(B(x,y)_t) \tag{2.17}$$

where $thin_2(e(x,y)_t) = 0$ if $N(B(x,y))_t - 1$ verifies the conditional set 2.15, 1 otherwise. The second check is also performed for all 8 neighboring elements of $B(x,y)$.

The event-based computation requires the use of a sliding time window. As with the event-based low pass filter, we implement it as decay of accumulated values, as defined in equation 2.18. The decay is applied to values of B so that previous computations do not affect events too far apart in time.

$$\frac{dB}{dt} = B(t-1) - \lambda(t) \tag{2.18}$$

where $B(t-1)$ is the previous value of B and $\lambda(t) = \alpha(t - t_{-1})/dt$.

$B(t)$ is the value of the mask B at time t, α is the convergence constant.

Unlike with the low-pass filter, the thinning algorithm cannot filter out events at the level of the local conditional checks. The order of the incoming events affects the computation. We solve this problem by implementing a successive filter that filters out events based on the state of the mask B. This filter is defined by equation 2.19

$$e_{out}(x,y)_t = \left\{ \begin{array}{c} e_{in}(x,y)_t \text{ if } B(x,y)_t = 1 \\ 0 \text{ if } B(x,y)_t = 0 \end{array} \right\} \tag{2.19}$$

where 0 means no event.

The pseudo-code algorithm is as follows:

```
// thinning on accumulator
float accumulator[][]
int mask[][]

for each event e
  // add grey level value to accumulator
  accumulator[e.x][e.y] = step * (type - 0.5);

  create mask by testing first set of conditions on neighbors of event
  for all i,j of pixels neighboring e.x,e.y
    mask[i][j] = test conditionsSet1;
  end for
  test second set of conditions on neighbors in mask accumulator
  for all i,j of pixels neighboring e.x,e.y
    mask[i][j] = test conditionsSet2;
  end for
end for

//generate events
for each event e
  if mask(e.x,e.y)==1
    generate event;
  end if
end for
```

The thinning algorithm is implemented within the jAER framework in the file net.sf.jaer.eventprocessing.filter.ThinningEventBased.java.

2.3.4 Results

With a DVS, we recorded the movement of an object, here a mug, see figure 2.13. We then spatially low pass filtered the data using our event-based filter described in previous section 2.2. The resulting data contains many thick segments, as shown in figure 2.14.
Figure 2.15 show the mask resulting from our event-based thinning filter. In a second loop, this mask is used to filter out events so than only events belonging to the thinned shape are transmitted past the filter.
In the following we applied the thinning algorithm to a recording of a moving H letter. An event-based low-pass filter is used to generates thick borders as shown in figure 2.16. We compare in figure 2.17 the results of applying the frame-based thinning algorithm of Zhang et al. [Zhang and Suen, 1984] and our event-based adaptation. The horizontal part of the letter H is not visible as the letter was moved horizontally.

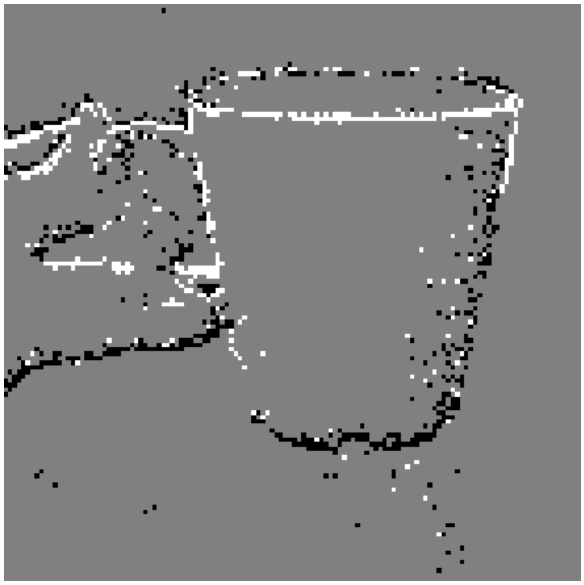

Figure 2.13: Raw data of a mug shaked left and right, accumulated over a 20ms time bin.

Figure 2.14: Spatially low-pass filtered record of a mug moved left and right, white dots represents ON events while black dots represents OFF events, accumulated over a 20ms time bin.

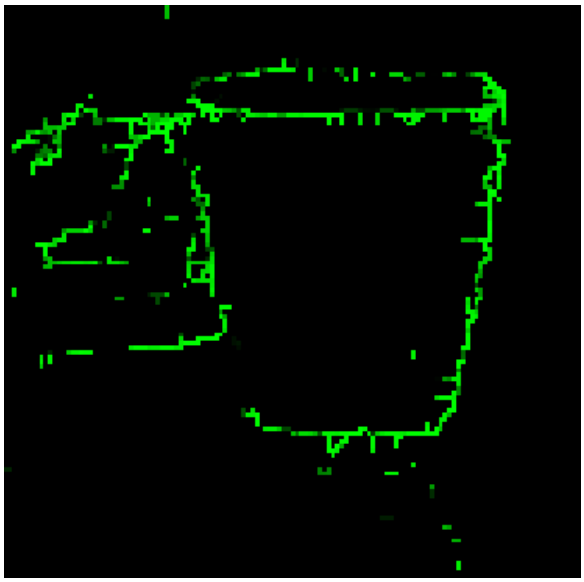

Figure 2.15: Thinned result of the spatially low-pass filtered record of a mug moved left and right, white dots represents ON events while black dots represents OFF events, with a decay time bin of 10ms.

Figure 2.16: Accumulation over 20ms of events. A recording of a letter, H, is made thick by the use of a low pass filter.

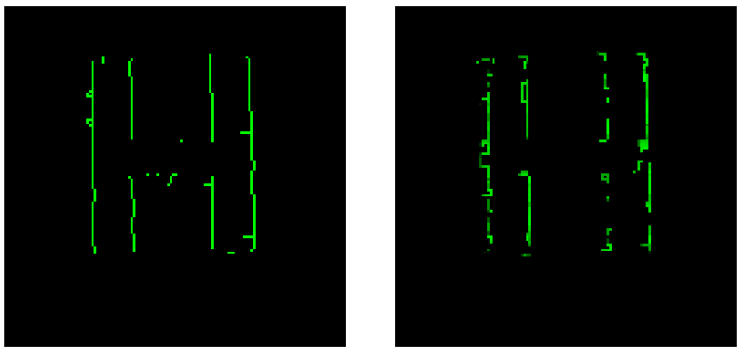

Figure 2.17: Thinning filters results for the moving letter H, a) frame-based thinning, b) event-based thinning

Limitations

The event-based algorithm uses a sliding time window and as with the low-pass filter algorithm, object motion within this time window perturbs the outcome of the filter. We show in figure 2.18 how this affects the event-based thinning filter. Moving points are wrongly detected as segments, breaking the shape of the thinned version of the moving object. Efficient sliding time windows are limited to the range [1ms,5ms].

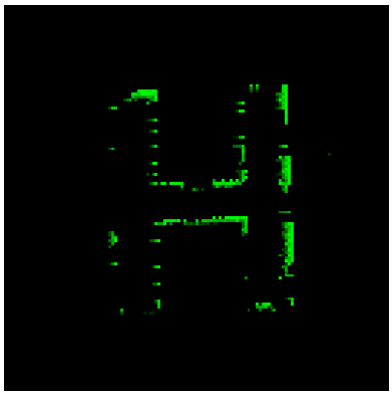

Figure 2.18: Result for the fast moving letter H using the event-based thinning algorithm

2.4 Conclusions

We showed how to adapt frame-based computer vision method to process visual data represented as events. These event-based algorithms are fully capable of performing spatial computation. The interaction between events is implemented through a sliding time window. The result is expressed as events, using a second generative loop when the order of events affects the computation. The algorithms exclusively use local computations. The spatial interactions occur within a time window exploiting the full temporal resolution of the events stream, but as a consequence, fast moving object will affect the validity of the filters result. It should also be noted that thinning algorithms do not appear useful when processing event-based data that represents contrast changes. Indeed, the sensitivity to contrast changes only is similar to the edges extraction process. But events can also represent sustained responses or higher-level features that can form thick representations.
In the next chapter we address a task in which not only the spatial information

but also the temporal information can be exploited, thus investigating the advantages of event-based representation over frame-based data in vision processing.

Bibliography

[Belbachir et al., 2007] Belbachir, A., Litzenberger, M., Posch, C., and Schon, P. (2007). Real-time vision using a smart sensor system. *IEEE International Symposium on Industrial Electronics*, pages 1968–1973.

[Davies, 2005] Davies, E. (2005). *Basic Image Filtering Operations*, chapter 3. Academic Press, 3 edition.

[Davies and Plummer, 1981] Davies, E. and Plummer, A. (1981). Thinning algorithms: a critique and a new methodology. *Pattern Recognition*, 14(1-6):53–63.

[Fukunaga et al., 1975] Fukunaga, K., , and Hostetler, L. D. (1975). The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition. *IEEE Transactions on Information Theory*, 21(1):32–40.

[Gonzalez and Woods, 2002] Gonzalez, R. and Woods, R. (2002). *Digital Image Processing*. Prentice Hall, 2 edition.

[Gu et al., 2004] Gu, X., Yu, D., and Zhang, L. (2004). Image thinning using pulse coupled neural network. *Pattern Recognition Letters*, Pattern Re(9):1075–1084.

[Lin and Chen, 1995] Lin, J.-Y. and Chen, Z. (1995). A Chinese-character thinning algorithm based on global features and contour information. *Pattern Recognition*, 28(4):493–512.

[Lindenbaum et al., 1994] Lindenbaum, M., Fischer, M., and Bruckstein, A. M. (1994). On Gabor contribution to image enhancement,. *Pattern Recognition*, 27:1–8.

[Litzenberger et al., 2006] Litzenberger, M., Posch, C., Bauer, D., Belbachir, A., Schon, P., Kohn, B., and Garn, H. (2006). Embedded vision system for real-time object tracking using an asynchronous transient vision sensor. *Digital Signal Processing Workshop, 12th - Signal Processing Education Workshop, 4th*, pages 173–178.

[Melhi et al., 2001] Melhi, M., Ipson, S. S., and Booth, W. (2001). A novel triangulation procedure for thinning hand-written text. *Pattern Recognition Letters*, 22(10):1059–1071.

[Parker, 2009] Parker, A. R. (2009). On the origin of optics, Article in Press. *Optics & Laser Technology*.

[Perez-Carrasco et al., 2008] Perez-Carrasco, J. A., Serrano, C., Acha, B., Serrano-Gotarredona, T., and Linares-Barranco, B. (2008). Event based vision sensing and processing. *2008 15th IEEE International Conference on Image Processing*, pages 1392–1395.

[Pratt, 1991] Pratt, W. K. (1991). *Digital Image Processing*. Wiley.

[Shang and Yi, 2007] Shang, L. and Yi, Z. (2007). A class of binary images thinning using two PCNNs. *Neurocomputing*, 70(4-6):1096–1101.

[Sherman, 1959] Sherman, H. (1959). A quasi-topological method for the recognition of line patterns. *Information Processing, Proc. UNESCO Conf.*, pages 232–238.

[Suen and Wang, 1994] Suen, C. Y. and Wang, P. S.-p. (1994). *Thinning methodologies for pattern recognition*. World Scientific, Singapor.

[Zhang and Suen, 1984] Zhang, T. Y. and Suen, C. Y. (1984). A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239.

[Zhang, 1997] Zhang, Y. (1997). Redundancy of parallel thinning. *Pattern Recognition Letters*, 18(1):27–35.

[Zhang and Wang, 1992] Zhang, Y. and Wang, P. (1992). Analysis of Thinning Algorithms. *IEEE Transactions on Image Processing*.

3

Event-based Stereo Vision: Calibration

In the previous chapter we have studied how to spatially process images using event-based algorithms, defining and implementing convolutions that operate in the spatio-temporal data space. In the following chapters we propose to extend our event-based framework to perform stereo vision, a challenging task requiring both spatial and temporal computations. Before we address how to obtain depth from stereo vision, the present chapter focuses on introducing the mandatory calibration process. The calibration is the measurement of all parameters that define the relationship between the three different elements of the stereo vision setup : the two vision sensors and the world. We first introduce the mathematical tools needed to manipulate the camera geometrical properties, then we follow the calibration procedure to ready our stereo vision setup.

3.1 Stereo vision and Projective Geometry

3.1.1 Definitions

Stereo vision is the process of using two vision sensors (cameras,eyes,...) to reconstruct a visual scene's structure in three dimension (3D). More generic ways of reconstructing the scene structure describe the use of multiple views (two or more) taken from different spatial positions either using multiple vision sensors or sequential captures. The principle stays the same in stereo vision, with the particularity that two vision sensors are used in a stereo configuration, side by side and with a similar orientation. Stereo vision allows to compute the 3D structure of the viewed scene based on the disparities between the two views. Multiple view's scene reconstruction is based on the differences, or disparities, in the obtained image representations of the scene. A physical object in the field of view of two cameras, or eyes, projects onto different positions in the cameras 2D image planes according to the depth of the object (its distance to the sensors). The depth of the object can be reconstructed by triangulation of the pixel responses and the

camera focal points. Its is thus important first to understand the spatial relationship between the cameras and the scene, and secondly the relationship between the two cameras pixels activities.

Projective geometry

The activity of the visual sensor is the result of the projection of the 3D scene structures onto the 2D sensor plane. Thus the shapes and visual properties of the objects in the scene are transformed and loosing one dimension when captured by the sensor. Projective geometry is the study and characterization of these transformations [Hartley and Zisserman, 2003]. In projective geometry, each point of n dimensions is the projection from a $n+1$ dimensional space through a projective vector whose origin defines the point of coordinate zero, see figure 3.1 for an example of a 2D projective space. The 2D space is represented in planes orthogonal to the projection dimension w.

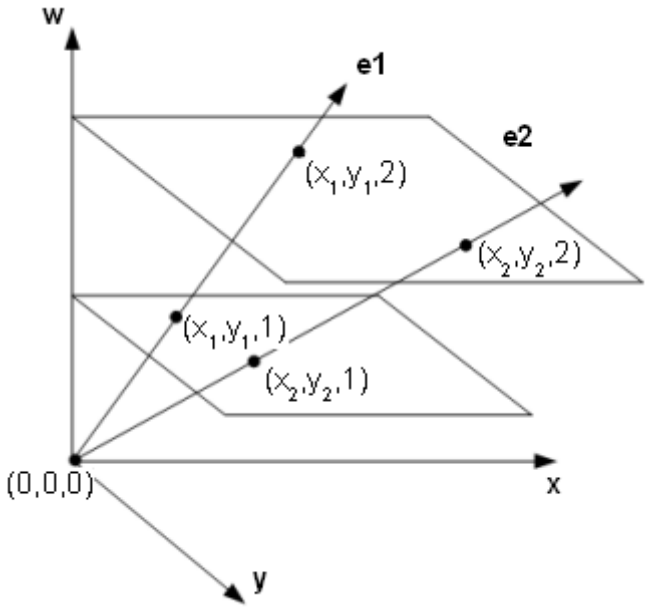

Figure 3.1: Projective Space. The 2D planes of $\mathbb{P}^2$ are all orthogonal to the projection dimension w. Here only the planes at $w = 1$ and $w = 2$ are shown.

In projective space $\mathbb{P}$ the points are described by homogeneous coordinates that include the additional w (or T in $\mathbb{P}^3$) term [Hartley and Zisserman, 2003]. So a point p in $\mathbb{P}^2$ is of homogeneous coordinates (x, y, w) as in figure 3.1. A point (X, Y, Z) in $\mathbb{R}^3$ is equivalent to $(X, Y, Z, 1)$ or more generically $(X/T, Y/T, Z/T, T)$

in $\mathbb{P}^3$. The use of homogeneous coordinates allow us to compute spatial transformations between different hyperplanes cutting the projection that defines the scene. Such transformations can describe the position and characteristics of the cameras.

3.1.2 Camera Matrix

The first step is to understand the spatial relationship between one camera and the scene. To characterize our camera, we define the 3D space of the scene as $\mathbb{P}^3$, equivalent to $\mathbb{R}^3$ along with points at infinity (defined as points with $T = 0$). The 2D plane of the camera view is $\mathbb{P}^2$. We model the lens of the camera as a point lens, assuming all rays of light reaching the pixels passes through a central point in the lens. The central projection that transforms $\mathbb{P}^3$ into $\mathbb{P}^2$ is a map associating any point (X, Y, Z, T) to (x, y, w). With the projection origin at $(0, 0, 0, 1)$, the result of the mapping is simply $x = X$, $y = Y$ and $w = Z$. This is described by the equation 3.1 : the multiplication of the point coordinates by a block matrix of dimension 3×4 composed of the identity matrix $I_{3\times3}$ and a zero filled vector O_3 to account for the dimension T.

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = P_{3\times4} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix} \tag{3.1}$$

with

$$P_{3\times4} = [I_{3\times3}|O_3] \tag{3.2}$$

$$I_{3\times3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, O_3 = (0\ 0\ 0) \tag{3.3}$$

In the generic case, the most general transformation from $\mathbb{P}^3$ into $\mathbb{P}^2$ is an arbitrary $P_{3\times4}$ matrix. This $P_{3\times4}$ matrix is known as the camera matrix.
Another useful tool in projective geometry is the homography H which is a 3×3 matrix describing the transformation from one intersecting plane to another. When all points lie in the same plane, $P_{3\times4}$ can be simplified to $H_{3\times3}$ by dropping the Z dimension as it will be the same for all points, as in equation 3.4

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = H_{3\times3} \begin{pmatrix} X \\ Y \\ T \end{pmatrix} \tag{3.4}$$

Now we have all the projective geometry tools needed to characterize our camera.

3.1.3 One Camera Projection

We model the camera as a simple pin-hole camera, where all rays of light focus through a single point, the camera center C, see figure 3.2. We choose C to be the origin in the euclidean coordinate system, and the plane Z=f to be the image plane. The principal point p is the center of the image plane and f is the focal length of the camera, the distance between the camera center and the principal point.

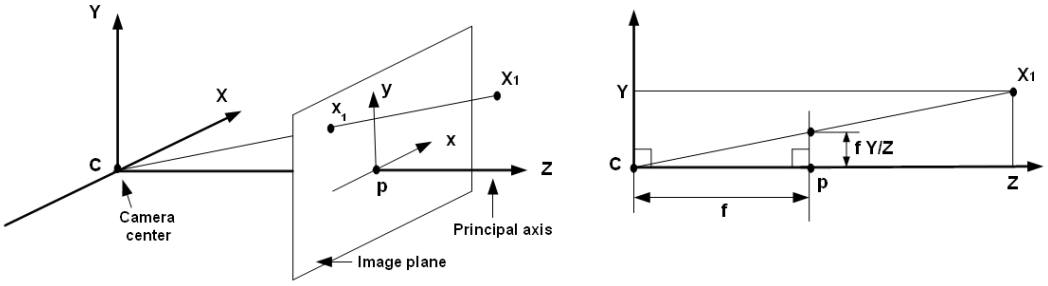

Figure 3.2: Pin-hole camera model, also viewed from the side, reproduced from [Hartley and Zisserman, 2003]

The projection of a point $(X, Y, Z)^T$ through C onto the image plane depends on the focal length f as is seen in 3.5 ($(X, Y, Z)^T$ denotes the transpose of the matrix (X, Y, Z)).

$$(X, Y, Z)^T \mapsto (fX/Z, fY/Z, f)^T \tag{3.5}$$

Using homogeneous coordinates we can rewrite this transformation 3.5 using the camera matrix $P_{3\times4}$, as 3.6

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX/Z \\ fY/Z \\ f \end{pmatrix} = P_{3\times4} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.6}$$

where

$$P_{3\times4} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \tag{3.7}$$

This is true assuming that the origin in the image frame coordinates system and in the camera system is the same. In practice we need to take into account the difference, as shown in figure 3.3.

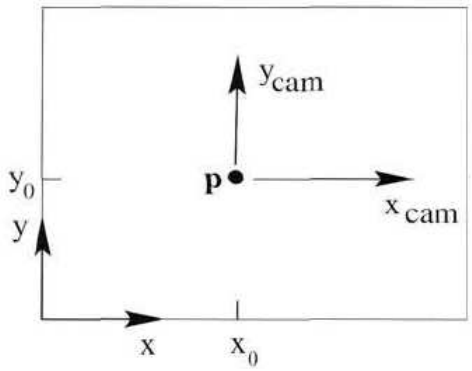

Figure 3.3: Differences between Camera and Image coordinate systems, reproduced from [Hartley and Zisserman, 2003]

As a consequence of this difference, the mapping is rewritten as in equation 3.8

$$(X, Y, Z)^T \mapsto (fX/Z + p_x, fY/Z, f + p_y)^T \tag{3.8}$$

where (p_x, p_y) are the coordinate of the principal point in the image coordinate system.
This is easily integrated into the homogeneous transformation by modifying the camera matrix P into K as in 3.9

$$K = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \tag{3.9}$$

The matrix K is called the camera calibration matrix. It defines the camera *intrinsic* properties: its focal length f and its principal point coordinates (p_x, p_y).
The concise form of the mapping can thus be written as in equation 3.10

$$x = K[I|O]X_{cam} \tag{3.10}$$

where x is a point in the image, K is the camera calibration matrix, I the identity matrix, O a zero 3-vector, and X_{cam} is a point in the real scene (or world) expressed in the camera's coordinates.
Indeed, points in the real world can have coordinates in different Euclidean systems, and in general we want to express them in the *world coordinate system* rather

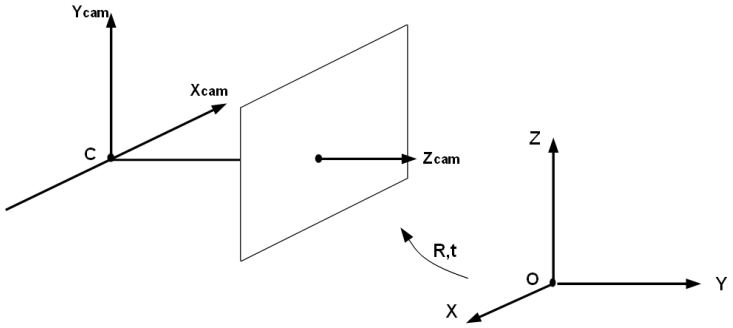

Figure 3.4: Relation between Camera and World coordinate systems, reproduced from [Hartley and Zisserman, 2003]

than the camera coordinate system. The two systems are related through translation and rotation as visualized in figure 3.4.
The rotation is described by the 3×3 matrix R representing the orientation of the camera, and the translation is defined by the 3-vector $\tilde{C}$ which represent the coordinates of the camera center in the world coordinate system. Then for a point $\tilde{X}$ in the world Euclidean coordinates, its corresponding point $\tilde{X}_{cam}$ in the camera Euclidean coordinate system is obtained from

$$\tilde{X}_{cam} = R(\tilde{C} - \tilde{X}) \tag{3.11}$$

which is best rewritten in homogeneous coordinates with the points X_{cam} and $\mathbf{X}$ as in 3.12

$$X_{cam} = \begin{bmatrix} R & -R\tilde{C} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} R & -R\tilde{C} \\ 0 & 1 \end{bmatrix} \mathbf{X} \tag{3.12}$$

We can then rewrite 3.10 to use the homogeneous world coordinate system as 3.13

$$x = KR[I| - \tilde{C}]\mathbf{X} \tag{3.13}$$

which by posing $t = -R\tilde{C}$ allow us to compute the camera matrix P as in 3.14

$$P = K[R| - t] \tag{3.14}$$

The rotation R and the translation $\tilde{C}$ are the *extrinsic* properties of the camera that relate it to the scene. The calibration of a camera consists of computing both its intrinsic parameters f and (p_x, p_y) and its extrinsic parameters R and $\tilde{C}$.

3.1.4 Computation of the Camera Matrix

To calibrate the camera and learn its intrinsic parameters we need to compute its camera matrix P. In order to do so we need to use a minimum amount of pre-established correspondences $X_i \leftrightarrow x_i$ between points in the real world (X_i) and points in the camera's image plane (x_i). This is usually achieved by capturing the image of a well characterized pattern or object. The obtained correspondences are described by $x_i = PX_i$ for all i, from which we have to compute the 3×4 matrix P. By expressing this equation as the vector product $x_i \times PX_i = 0$, Hartley et al. [Hartley and Zisserman, 2003] derived a linear solution by posing equation 3.15

$$PX_i = \begin{pmatrix} p_1^T X_i \\ p_2^T X_i \\ p_3^T X_i \end{pmatrix} \tag{3.15}$$

and with $x_i = (x_i, y_i, w_i)^T$, the vector product becomes as described by 3.16

$$x_i \times PX_i = \begin{pmatrix} y_i p_3^T X_i - w_i p_2^T X_i \\ w_i p_1^T X_i - x_i p_3^T X_i \\ x_i p_2^T X_i - y_i p_1^T X_i \end{pmatrix} \tag{3.16}$$

This gives us three equations of the form $A_i p = 0$ in 3.17

$$A \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{bmatrix} 0^T & -w_i X_i^T & y_i X_i^T \\ w_i X_i^T & 0^T & -x_i X_i^T \\ -y_i X_i^T & x_i X_i^T & 0^T \end{bmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = 0 \tag{3.17}$$

A_i is the i^{th} row of matrix A and each $A_i p = 0$ correspond to an equation. Because only two of these equations are independent and the third can be computed from the two others, the number of equations to solve reduces to two in 3.18 [Sutherland, 1963] [Hartley and Zisserman, 2003] :

$$\begin{bmatrix} 0^T & -w_i X_i^T & y_i X_i^T \\ w_i X_i^T & 0^T & -x_i X_i^T \end{bmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = 0 \tag{3.18}$$

If the correspondences were exact, the linear equations $A_i p = 0$ would give an exact solution. But as the correspondences are bound to be noisy, we can only estimate the solution vector p as best as possible by minimizing a cost function. The usual cost functions used are algebraic or geometric distances. The Direct Linear Transformation algorithm (DLT) that estimate P [Hartley and Zisserman, 2003] minimizes the algebraic norm $\|Ap\|$. The geometric distance is measured either from reconstruction after estimation and comparison with the known points, or by symmetric estimation when the transformation is an homography.

In the case of the camera calibration, it is best to solve these equations using established algorithms such as proposed by Zhang [Zhang, 1999],
Heikkila [Heikkila and Silven, 1997], Bouguet [Bouguet, 2008] or Hartley et al. [Hartley and Zisserman, 2003].

Now that we can calibrate each camera independently, we can address the problem of relating the two cameras to each others.

3.1.5 Two Cameras

In stereo vision, the addition of a second camera aims at enabling the inverse mapping from $\mathbb{P}^2$ to $\mathbb{P}^3$ and thus reconstructing the 3D scene structure. In order to compute this transformation, we need to calibrate both cameras to find out their intrinsic and extrinsic parameters, but also to determine the relationship between the two cameras' image planes.

The relation to compute is the homography F that transforms points x_1 of the first camera into points x_2 in the second image plane when they both correspond to a real point X of the 3D world, see figure 3.5.

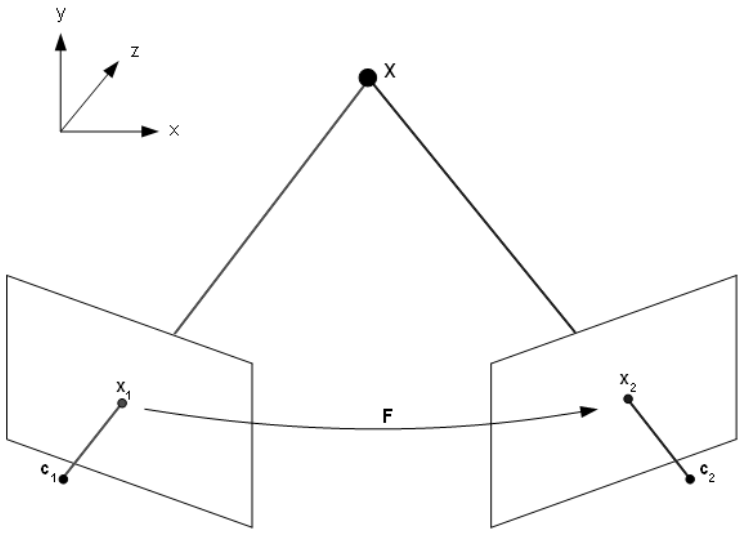

Figure 3.5: Corresponding points in both images correspond to a same point in real world and the transformation F is the relation between the two image plane corresponding points

3.1.6 Epipolar planes

Corresponding points x_1, x_2 and X all lie in the same plane, to which the camera centers also belongs. This plane is the epipolar plane. The baseline connecting the two camera centers belongs to this plane and cut each image plane in points called epipoles, e_1 and e_2 as in figure 3.6.

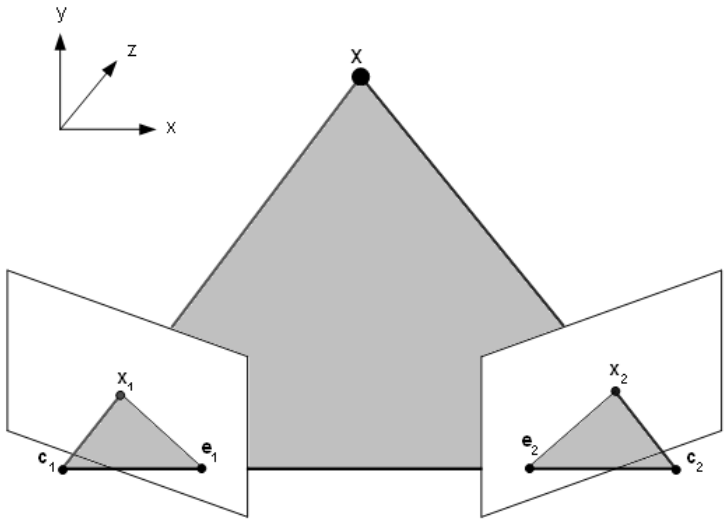

Figure 3.6: Corresponding points in both image plane and in the real world belong to the same plane, the epipolar plane

In the same way than the projection of a real world point X is x in the image plane, the back projection from x to X is a line L in the world 3D space. This line projects itself onto the second image plane as l_2. l_2 is called the epilolar line and represents all possible x_2 projections of X in the second image. The line l_2 links the epipole e_2 and the corresponding point x_1 in the other image plane. Thus the corresponding point x_2 of x_1 can only lie on the epipolar line l_2 and the transformation from one image to the other obeys the mapping $x_1 \mapsto l_2$.

3.1.7 Fundamental matrix

This mapping is represented by a 3×3 matrix, the *Fundamental matrix* F. Given a plane in the real world with X lying on it, the transformation as in figure 3.7 from x_1 to x_2 can be written as $x_2 = Hx_1$ and $l_2 = [e_2] \times x_2 = [e_2] \times Hx_1 = Fx_1$ where the fundamental matrix F is defined by 3.19, see figure 3.7.

$$F = [e_2] \times H \tag{3.19}$$

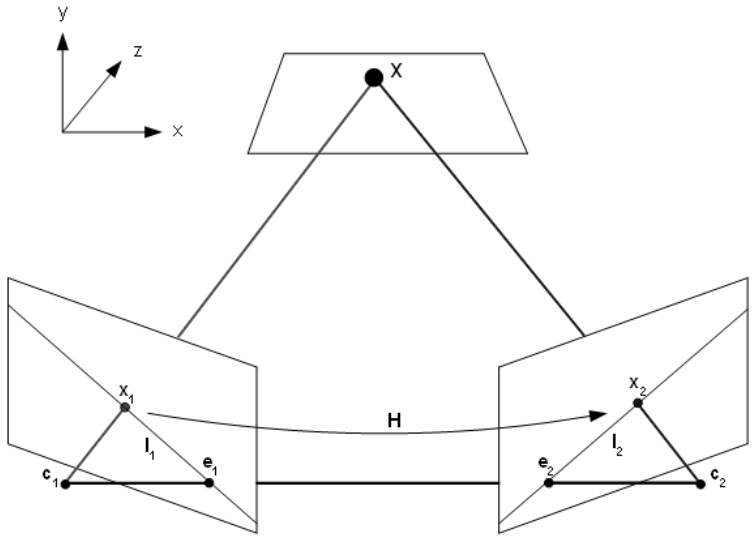

Figure 3.7: Computing F is done through evaluating H as the relation between the two pair of corresponding points for a given real world plane

The fundamental matrix F is thus the transformation from x_1 to x_2 that satisfies 3.20

$$x_2^T F x_1 = 0 \tag{3.20}$$

3.1.8 Computation of the Fundamental matrix

The equation 3.20 can be used to compute F from pairs of $x_1 \mapsto x_2$ point correspondences. The minimum amount of pairs is 7 [Hartley and Zisserman, 2003]. Having $x_1 = (x, y, 1)^T$ and $x_2 = (x', y', 1)^T$ gives us for each pair a linear equation involving unknown values of F. For a specific pair $(x, y, 1)$ with $(x', y', 1)$ this is defined by equation 3.21

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0 \tag{3.21}$$

with

$$F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \tag{3.22}$$

With f as a 9-vector representing the matrix F in row-major order this simply becomes

$$(x'x, x'y, x', y'x, y'y, y', x, y, 1)f = 0 \tag{3.23}$$

This gives us a set of equations on the form of 3.24

$$Af = \begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} f = 0 \tag{3.24}$$

The aim of the algorithm is then to find f by solving 3.24. Because the pair correspondence data is likely noisy, the result will be a least-squares solution corresponding to the smallest singular value of A obtained by singular value decomposition (SVD).
This is the basis of the simplest method for computing the fundamental matrix F, namely the 8-point algorithm [Longuet-Higgins, 1981]. Its uses 8 pair of $x_1 \mapsto x_2$ point correspondences. It is necessary to first normalize the data by translating the images' principal points to the origin to ensure a better stability of the algorithm [Hartley, 1997] [Hartley and Zisserman, 2003], simply by substracting the principal point camera coordinates from each image point coordinates. The algorithm is described in great details in [Hartley and Zisserman, 2003].
The calibration of the stereo rig of cameras thus implies to compute P_1, P_2 and F from a set of known points. We address the actual calibration procedure in the next section.

3.2 Calibration of the Event-based camera

The calibration process requires first to obtain a set of corresponding points. Instead of exploiting the properties of the spatio-temporal stream of data from the event-based camera, we first simplify the task by generating frames on which classical frame-based calibration algorithms can be executed.

3.2.1 Methods

Hardware Setup

We use two Dynamic Vision Sensors (DVS) [Lichtsteiner et al., 2008] of 128 $\times$ 128 pixels in a stereo rig, see figure 3.8. An additional synchronization board is used to provide a unique clock for the timing of events in both DVS. The lenses are 10 to 30 mm focal length set on maximum zoom at 30mm. The two DVS are directly

connected to the synchronization board (developed by Patrick Lichsteiner) which then output a single stream of events. Each events carry the label of its originating camera, 0 for the left camera and 1 for the right camera. The synchronization board is then connected by USB (Universal Serial Bus) to a Pentium 4 laptop.

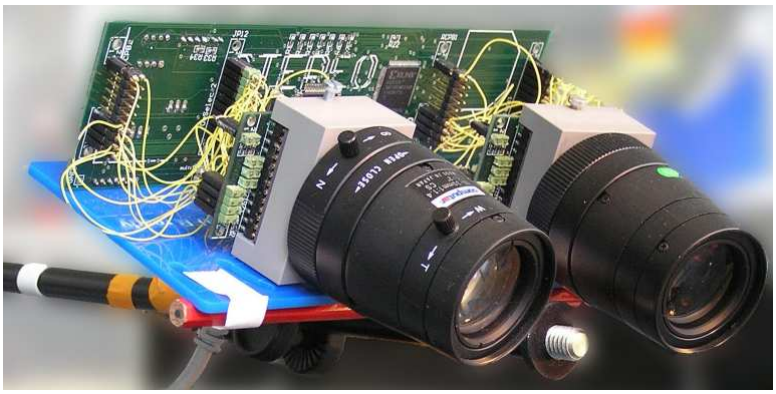

Figure 3.8: Two DVS 128 mounted in stereo and with an additional synchronizing capture board

Bouguet's calibration toolbox

We use the framework developed under Matlab by Bouguet [Bouguet, 2008] to calibrate a stereo rig of cameras. This algorithm facilitate the generation of corresponding reference points by automatically capturing corner points in pictures of a checkered pattern.
Static pictures of the pattern in different positions are obtained for each DVS by using an external shutter (see figure 3.9). The shutter is a piece of homogeneous plastic which is quickly moved in front of the lens, then the accumulation of events over the short transition phase results in a image of the static background scene. This methods allow to calibrate the DVS as if it was frame-based, and thus use previously developed calibration technique, but does not take advantage of the event data representation.
A minimum amount of 7 different views is required to calibrate a camera. We generated 10 pictures for each camera, from which 35 points are automatically extracted, see figure 3.10.

Zhang's estimation of the camera matrix

Bouguet's method implements Zhang's algorithm for the estimation of the camera matrix P [Zhang, 1999]. This algorithm computes the calibration matrix K

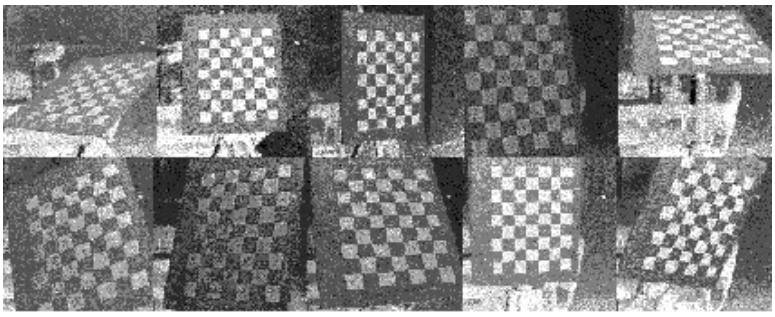

Figure 3.9: DVS Snapshots of the pattern in different positions used for calibration

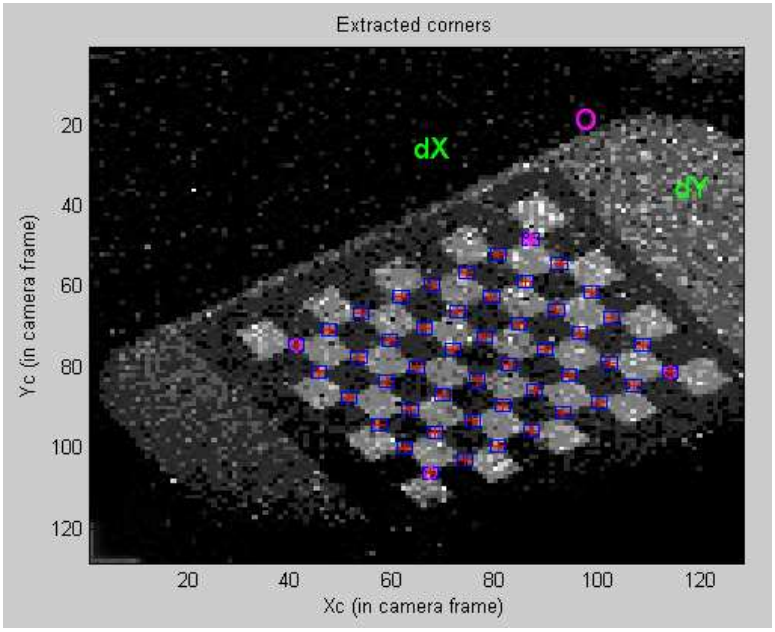

Figure 3.10: Automatically extracted corner points, in purple

(equation 3.25) by first finding the homography between the image and some model planes. Using views of checkered patterns, a model plane is the plane of the checkered pattern in the real world. Thus this method allows to calibrate the camera from few different views of real world planes. Bouguet's implementation is slightly different from Zhang in that distortion coefficients are processed differently : they are not computed at initialization phase and Bouguet uses the intrinsic model of Heikkila [Heikkila and Silven, 1997] that includes additional distortion coefficients. The distortion take into account here are the radial distortion in the image due to the lens geometry, and the tangential distortion due to the shift of the camera's central point. The distortion parameters are stored in a 5x1 vector (3 parameters for radial distortion and 2 for tangential distortion)

denoted kc in table 3.1.
The aim is to compute K

$$K = \begin{pmatrix} \alpha & c & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.25}$$

where α and β are possible scale factors related to the focal length, c is the skew of the u and v axis of the image plane as the camera design might make them not perfectly orthogonal, and u_0 and v_0 are the coordinates of the principal point of the camera in camera coordinates.
Equations 3.13 and 3.14 give

$$x = K[Rt]X \tag{3.26}$$

The homography between the image plane (in the image coordinate system) and its model plane in the real world coordinate system is defined by posing $Z = 0$ for the model plane. The homography H is thus

$$H = K \begin{bmatrix} R_{11} & R_{21} & t_1 \\ R_{12} & R_{22} & t_2 \\ R_{13} & R_{23} & t_3 \end{bmatrix} \tag{3.27}$$

This homography is then estimated by solving equation 3.16 for H. Because the data points and planes that are automatically extracted and noisy, Zhang [Zhang99] searches a maximum likelihood estimation of H by minimizing

$$\sum_i \|(x_i - \hat{x}_i)\|^2 \tag{3.28}$$

where

$$\hat{x}_i = \frac{1}{(H_{31}H_{32}H_{33})^T} X_i \begin{bmatrix} (H_{11}H_{12}H_{13})^T & X_i \\ (H_{21}H_{22}H_{23})^T & X_i \end{bmatrix} \tag{3.29}$$

This is a non-linear minimization that can be solved using the Levenberg-Marquardt algorithm, see [More, 1977], [Zhang, 1999].
Once the homography H is estimated, Zhang proposes to find K by solving the constraints 3.30 and 3.31

$$h_1^T (K^-1)^T K^-1 h_2 = 0 \tag{3.30}$$

$$h_1^T (K^-1)^T K^-1 h_1 = h_2^T (K^-1)^T K^-1 h_2 \tag{3.31}$$

So he poses

$$B = K^{-T}K^{-1} \equiv \begin{bmatrix} B_{11} & B_{21} & B_{31} \\ B_{12} & B_{22} & B_{32} \\ B_{13} & B_{23} & B_{33} \end{bmatrix} \tag{3.32}$$

that can be simplified as a vector b because B is symmetric,

$$b = (B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33})^T \tag{3.33}$$

Knowing our estimated H with its column vectors $h_i = [h_{i1}, h_{i2}, h_{i3}]^T$ it is derived that

$$h_i^T B h_j = v_i^T b \tag{3.34}$$

with $v_{ij} = [h_{i1}h_{j1}, h_{i1}h_{j1}]$.
So the constraints 3.30 and 3.31 are rewritten as

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} \tag{3.35}$$

Now by having n images of the model plane we can stack n equations (3.35) so that

$$VB = 0 \tag{3.36}$$

where V is a $2n \times 6$ matrix. With $n > 3$ there should be a unique solution b. Once b is found, the intrinsic parameters of the calibration matrix K can be computed from $B = \lambda(K^{-}1)^T K$ with λ an arbitrary scale factor. The intrinsic parameters are then computed from 3.37

$$\begin{aligned} v_0 &= (B_{12}B_{13} - B_{11}B_{23})/(B_{11}B_{22} - B_{12}^2) \\ \lambda &= B_{33} - [B_{13}^2 + v_0(B_{12}B_{13} - B_{11}B_{23}]/B_{11} \\ \alpha &= \sqrt{\lambda/B_{11}} \\ \beta &= \sqrt{\lambda B_{11}/(B_{11}B_{22} - B_{12}^2)} \\ c &= -B_{12}\alpha^2\beta/\lambda \\ u_0 &= cv_0/\alpha - B_{13}\alpha^2/\lambda \end{aligned} \tag{3.37}$$

Bouguet's mono calibration procedure

The above computations are all implemented by Bouguet in his toolbox framework [Bouguet, 2008]. The user need only provide the checkered pattern pictures and define the number of points in x and y on the checkered board, and the size in mm of each square of the checkered board. To automatically extract the corners we set a search window size of 3×3 pixels. For each image the user must

interactively indicate the boundaries of the checkered board by clicking on its four corners. The extrinsic parameters are then computed from the picture of a checkered board in the plane $Z = 0$.

Bouguet's stereo calibration procedure

When both cameras have been independently calibrated, all parameters are recomputed with stereo pairs of images of the checkered board in different positions inside the two views. The previously computed results serve as initial values for this global stereo optimization of the camera parameters. Extrinsic parameters defining the relationship between the left and right camera (rotation and translation) are also computed.
The matlab code we used and a read me file can be found in the jAER repository under /host/matlab/ratmonitoring/calibration.

3.2.2 Results

We present here the calibration results obtained by using the adaptation of Bouguet's calibration methods to our event-based DVS stereo rig.

Mono calibration

We first calibrated each camera independently. The computation of the extrinsic parameters of one camera allows to reconstruct the real world position of the checkered patterns that were presented to it. The reconstruction for the right camera is visible in figure 3.11 and figure 3.12 from the side.
We obtained the intrinsic parameters of both cameras independently. The result for the left camera is shown in table 3.1

Focal Length:	fc = [648.37734 688.35200] ± [134.21271 157.21245]
Principal point:	cc = [63.50000 63.50000] ± [0.00000 0.00000]
Skew:	alpha_c = [0.00000] ± [0.00000]
Distortion:	kc = [-3.09891 251.13677 0.05162 0.00952 0.00000]
	± [4.22135 507.23148 0.06838 0.04402 0.00000]
Pixel error:	err = [0.34649 0.35438]

Table 3.1: Intrinsic parameters of left camera

The focal Length fc correspond to α and β of K in equation 3.25. The unit is number of horinzontal and vertical pixels. To obtain the focal length in mm we have to multiply it by the size of a pixel. With the DVS128 the pixel measure 0.04

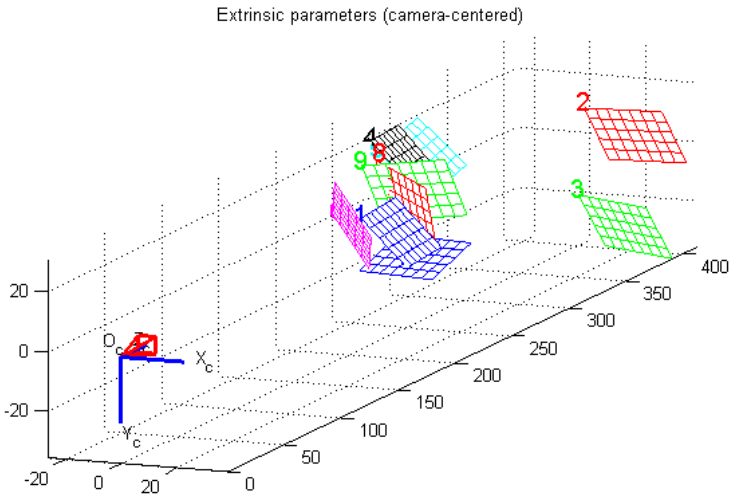

Figure 3.11: Reconstruction of the calibration patterns (labelled 1 to 9) for the right camera.

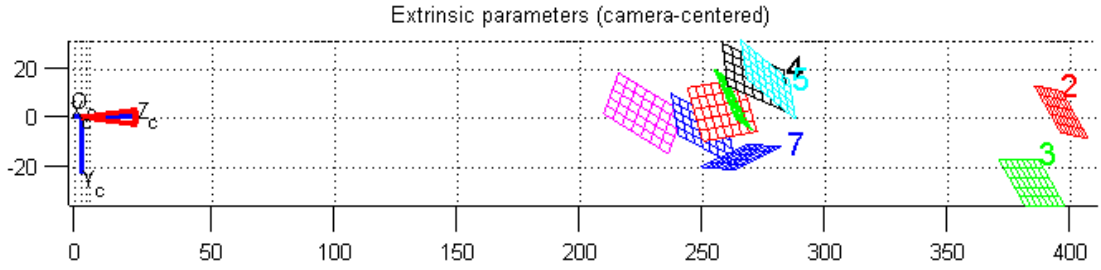

Figure 3.12: Reconstruction of the calibration patterns (labelled 1 to 9)for the right camera, view from the side.

mm so $fc(\alpha)$ = 648.37734 $\times$ 0.04 = 25.92mm. This is in the range of the used lens (10-30mm) but not so close to the value we set (30mm) using the lens ring. The principal point is given in pixels in the image plane coordinates. The estimation of the principal point can be a problem later when we combine the calibration of both cameras, so we disabled its estimation and set it to its default central value cc = [63.50000 63.50000]. Similarly, we explicitly did not compute the skew. The pixel error is below one pixel, giving the calibration a subpixel accuracy.
The obtained intrinsic parameters for the right camera are shown in table 3.2
The pixel error is computed by reprojection, see figure 3.13
The results are similar for the right camera. The principal point and the skew

Focal Length:	fc = [462.35170 494.92099] ± [76.00206 87.92303]
Principal point:	cc = [63.50000 63.50000] ± [0.00000 0.00000]
Skew:	alpha_c = [0.00000] ± [0.00000]
Distortion:	kc = [3.00460 -147.01155 0.21287 -0.02332 0.00000]
	± [2.57345 149.22773 0.05982 0.04293 0.00000]
Pixel error:	err = [0.34649 0.35438]

Table 3.2: Intrinsic parameters of right camera

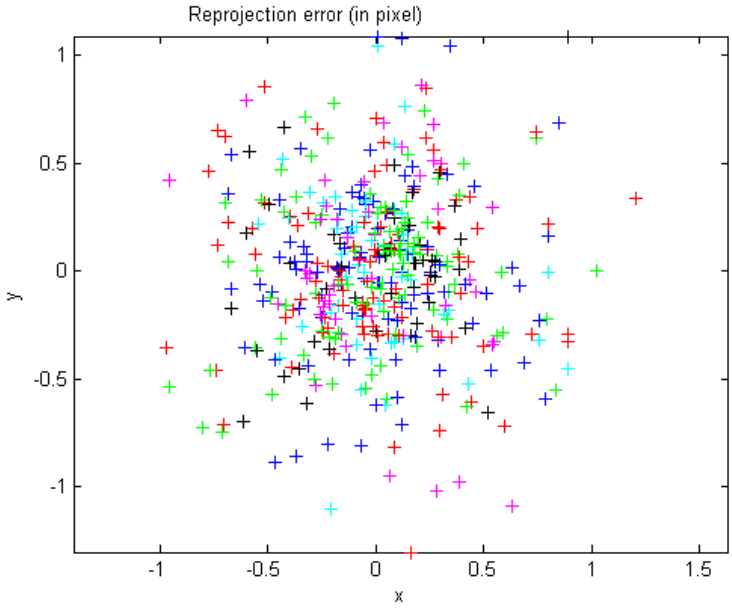

Figure 3.13: Reprojection of pixel errors, each cross is a reprojection and closer to (0,0) is more accurate

have not been estimated here, as with the left camera. Here the computed focal length is of the order of 462.35 × 0.04 = 18.494 mm, which is far from the 30mm set. This is possibly due to the noise in the selection of points in the image. As we will see in the next section on stereo calibration, the following optimization will attempt to correct these errors.

Stereo calibration

A global stereo optimization is performed on the previously computed calibration results, this time with stereo pairs of images of the same checkered pattern positions. All intrinsic and extrinsic parameters are recomputed to minimize the reprojection errors on both cameras for all checkered patterns. The uncertainties

on the intrinsic parameters is reduced as the global stereo optimization is performed over a minimal set of unknown parameters [Bouguet, 2008]. In particular, only 6 degrees of freedom are considered for the location of the checkered pattern for each stereo pair. This increases the global rigidity of the transformation from the left image plane to the right image plane. The new results are shown in tables 3.3 and 3.4

Focal Length:	fc = [792.91796 815.16570] ± [45.34433 47.20273]
Principal point:	cc = [63.50000 63.50000] ± [0.00000 0.00000]
Skew:	alpha_c = [0.00000] ± [0.00000]
Distortion:	kc = [1.80529 -26.58328 0.02296 -0.00575 0.00000]
	± [7.17984 1276.06013 0.03974 0.04106 0.00000]

Table 3.3: Intrinsic parameters of left camera after stereo calibration

The optimized focal length is now estimated as 792.9 × 0.04 = 31.71mm, much closer to the indicative value of the lens focal length (30mm) than before optimization.

Focal Length:	fc = [745.83544 778.93446] ± [41.33831 46.34561]
Principal point:	cc = [63.50000 63.50000] ± [0.00000 0.00000]
Skew:	alpha_c = [0.00000] ± [0.00000]
Distortion:	kc = [1.71350 -239.32347 0.04347 -0.00676 0.00000]
	± [5.43930 736.07686 0.03941 0.04644 0.00000]

Table 3.4: Intrinsic parameters of right camera after stereo calibration

The estimation is also better for the right camera. Its focal length is now estimated as 745.83 × 0.04 = 29.83mm.

Rotation vector:	om = [0.00178 0.21112 0.03127] ± [0.00081 0.00987 0.00474]
Translation vector:	T = [-95.51996 -2.33898 5.22091] ± [2.68993 0.43598 14.70869]

Table 3.5: Extrinsic parameters relating the right camera to the left camera

The extrinsic parameters of the stereo rig (table 3.5) give the relationship between the left and right cameras. The translation T is given in mm. The rotation om is computed here as a vector. We obtain the rotation matrix R from $om = (v_1 v_2 v_3)$ by using the Rodrigues rotation formula which gives us R as in equation 3.38.

$$R = \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{pmatrix} \tag{3.38}$$

The reprojection of the checkered patterns after calibration is shown in figure 3.14.

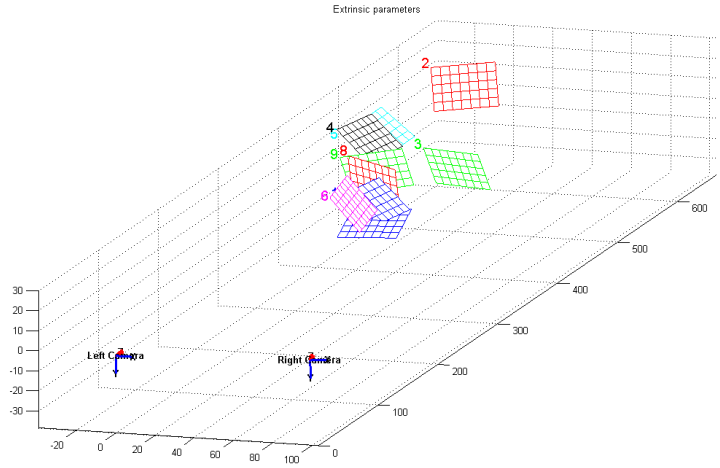

Figure 3.14: Reconstruction of the stereo rig and the calibration patterns

The obtained calibration results for the two cameras can be compared against the real measurements. The calibrated focal lengths (31.71mm for the left camera and 29.83mm for the right camera) are likely more accurate than the lens setting of 30mm due to manufacturing accuracy. The translation $T = (-95.51996 - 2.338985.22091)$ between the two lenses, gives a distance of as $\|T\| = 95.6911$ mm. This distance was manually measured as 94 mm. This results are likely more accurate than our manual measurements, and can give additional information on distortion and internal camera parameters such as the position of the principal point, which cannot be easily measured.

Fundamental matrix

We can compute the fundamental matrix F, describing the relation between the two cameras planes, from the results obtained previously. This fundamental matrix can be later used to recompute epipolar lines. Now that we have R, T and the camera matrices C_1 for the left camera and C_2 for the right camera, defined as in equation 3.39 from the values fc and cc, we can compute F from equation 3.40

$$C = \begin{bmatrix} fc(1) & 0 & cc(1) \\ 0 & fc(2) & cc(2) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.39}$$

where $fc(1)$ and $fc(2)$ are computed focal length for the camera, and $cc(1)$ and $cc(2)$ are x,y coordinates of the camera central point.

$$F = (C_2^{-1})^T * E * C_1^{-1} \tag{3.40}$$

where $E = R * S$ is the essential matrix and S is defined as in equation 3.41

$$S = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \tag{3.41}$$

where T is the translation obtained from calibration, defined as $T = [T_x T_y T_z]$.

3.3 Epipolar Rectification

Knowing the intrinsic and extrinsic parameters of the camera, we are almost ready to perform stereo vision. The last problem is a matter of optimization. The reconstruction of the 3D scene from the two vision sensors will involve comparing points belonging to the same epipolar lines. But the epipolar planes for each point x_i vary with the position of x_i, rotating around the baseline linking the two cameras centers. Their intersections with the image planes, the epipolar lines, have to be computed for each point. Instead of computing the epipolar line position for each x_i point of the image, it is possible to distort the image to rectify the epipolar lines into lines parallel to the x axis. This would improve dramatically any search algorithm for point correspondences on epipolar lines.

3.3.1 Definition

The epipolar rectification transforms the two image planes into coplanar planes so that all pairs of corresponding epipolar lines become colinear, as described in figure 3.15. This is equivalent to physically moving the cameras so that they become aligned side by side and facing in exactly the same orientation.
For one camera, the rectification is a transformation H_r that maps the original camera matrix P_o to a new camera matrix P_n

$$H_r = P_o P_n^- 1 \tag{3.42}$$

The epipole of a camera, denoted e (e_1 and e_2 in figure 3.15), is the point at the intersection of the epipolar line of the principal point of the camera and the imaginary line connecting the two cameras' centers (c_1 and c_2 in figure 3.15) in a stereo setup. When the two image planes from left and right cameras are coplanar,

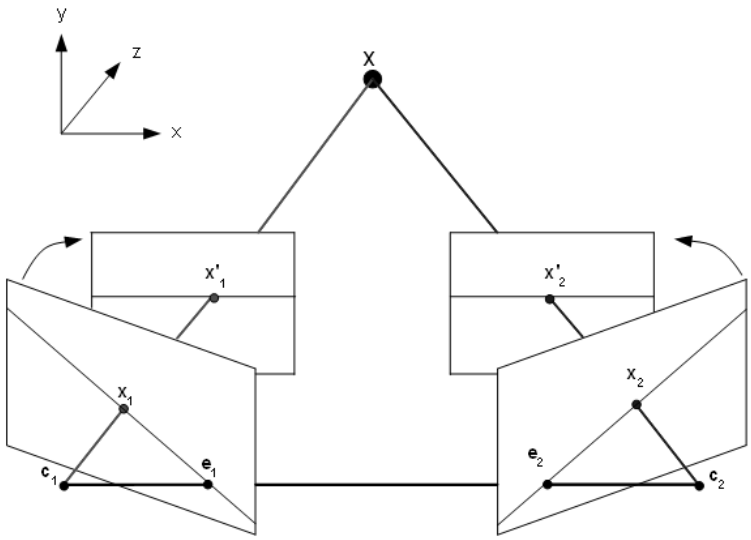

Figure 3.15: Epipolar rectification transforms the image planes so that they end up coplanar

the epipole e lies at infinity. So H_r is the transformation that moves e to the homogeneous coordinates $(1,0,0)^T$. If e is of coordinates $(f,0,1)^T$, Hartley et al. [Hartley and Zisserman, 2003] give the transformation G as in 3.43

$$G = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/f & 0 & 1 \end{pmatrix} \tag{3.43}$$

that moves the point e to $(f,0,0)^T$, see figure 3.17.
But for the general case this is done in three steps : first having the principal point of the camera at the origin by translating it if needed using the vector T. Secondly we rotate the image plane around its principal axis (linking the camera center and the principal point) so that the epipole lie on the x axis with coordinate $(f,0,1)^T$, see figure 3.16. Knowing the fundamental matrix F and thus the location of the epipole e, we can apply the transformation $H_1 = GRT$ to obtain a rectified image plane for one camera.
The matching transformation H_2 that maps the other camera of the stereo pair to a coplanar plane is found by minimizing the least-square distance $\sum_i d(H_1 x_{1_i} H_2 x_{2_i})$

3.3.2 Methods

The stereo calibration tool provided by Bouguet [Bouguet, 2008] automatically computes the fundamental matrix F and the epipole's location. Its also provides

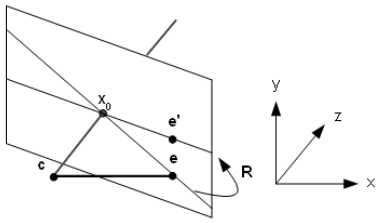

Figure 3.16: Rotation of the epipole prior to epipolar rectification

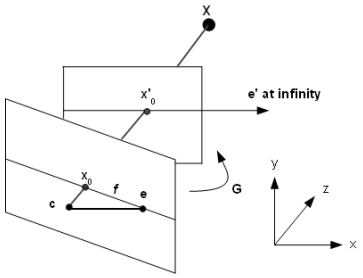

Figure 3.17: Epipolar rectification by moving the epipole to infinity

a rectification function that operates as described above. We have modified the algorithm of image rectification so that instead of generating corrected images, it creates a map between original pixel coordinates and new rectified pixel coordinates. Because of the low spatial resolution of the camera, there is a need for subpixel accuracy during the rectification, otherwise some pixel information may disappear. This is solved by extrapolating neighboring pixels with a 2×2 kernel. The result is a 2×2 map of factors for each rectified pixel. The result of the epipolar rectification without extrapolation would lead to loosing some pixel values, as can be seen in figure 3.18, respectively for the left camera in a), and for the right in b).
Using a kernel for extrapolating the neighboring pixel values is not applicable in the case of event-based camera, where the data is represented as sparse temporal contrast events and not as a intensity gradients. We solved this problem by defining the pixel transformation using the mapping factors that form 2×2 index maps between original pixels and rectified pixels where original pixels can map to more than one rectified pixel. We then randomly associate incoming events to one of their 2×2 map of target pixels. The process is optimum when the target map coefficients are used as weight for the random choices. Figure 3.19 highlights the difference between rectifications without and with extrapolation. The extrapolation is critical in order to use the spatial neighborhood information.
We then follow the same calibration procedure as for the stereo calibration with the additional image rectification.

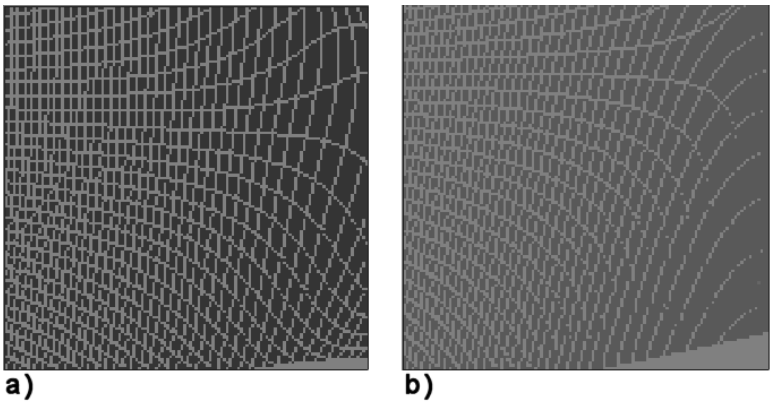

Figure 3.18: Epipolar rectification requires extrapolation. In this picture, dark grey points are points whose value is carried by the rectification. The light grey points are missing values. A grid of missing points is visible when no extrapolation is done when rectifying event-based data, but we solves this by randomly switching between multiple associations. a) left camera, b) right camera.

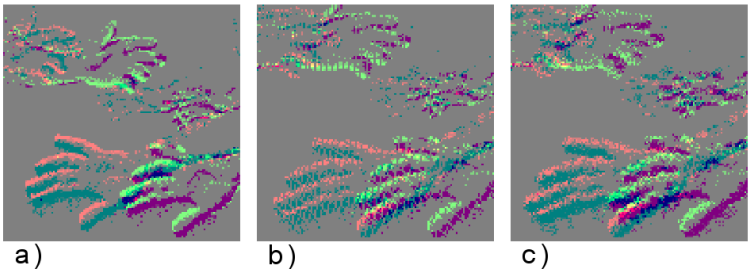

Figure 3.19: a) Accumulation over 40ms in a recording of three moving hands, b) epipolar rectification without extrapolation, c) epipolar rectification with event-based extrapolation. For the left camera, ON events are green and OFF events are purple while for the right camera, ON events are salmon and OFF events are cyan. Notice how the two (left and right) representations of the hand in the top left are shifted (in (b) and (c)) so that they lay on the same horizontal line. The extrapolation in (c) is visible as the hands' contours are not jagged by missing values anymore.

3.3.3 Results

The generated index maps allow to modify on-line the image coordinates of incoming events so that epipolar lines in both image planes become parallel to the x-axis. We have accumulated events so as to create a frame of a checkered board for both views before rectification, see figure 3.20. A red horizontal line has been drawn to highlight the non alignment of the patterns.

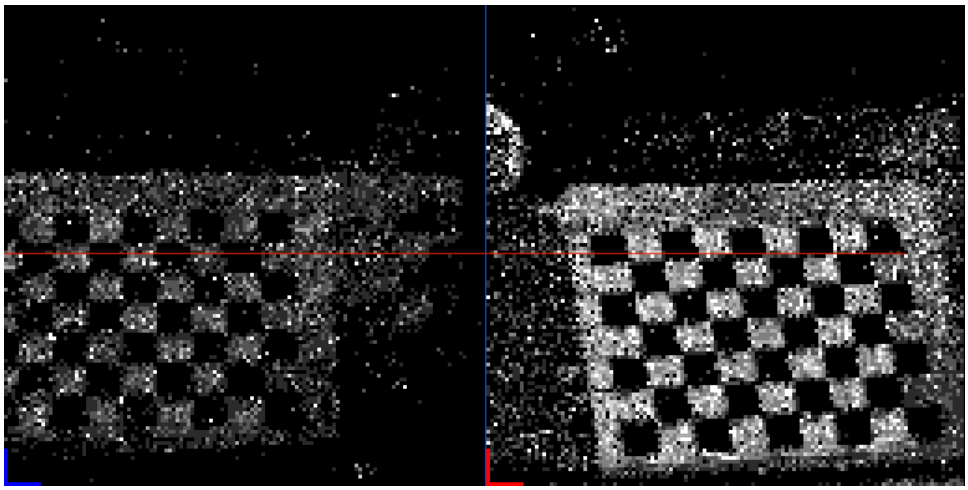

Figure 3.20: Comparison of the view of a checkered board in both camera before rectification. A red horizontal line has been drawn to highlight the non alignment of the patterns.

The result of the transformation is visible in figure 3.21. The red horizontal line shows the improvement of the alignment over the previously non rectified example.

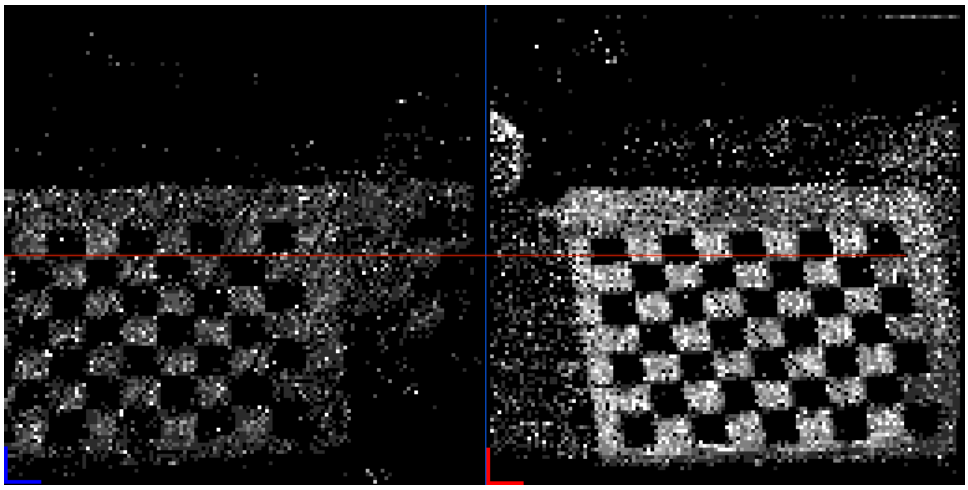

Figure 3.21: Comparison of the view of a checkered board in both camera after rectification. The red horizontal line shows the improvement of the alignment over the previously non rectified example.

The same comparison on raw data (accumulating in 20ms time window for display purpose) from the stereo rig of event-based DVS can be seen in figure 3.22.

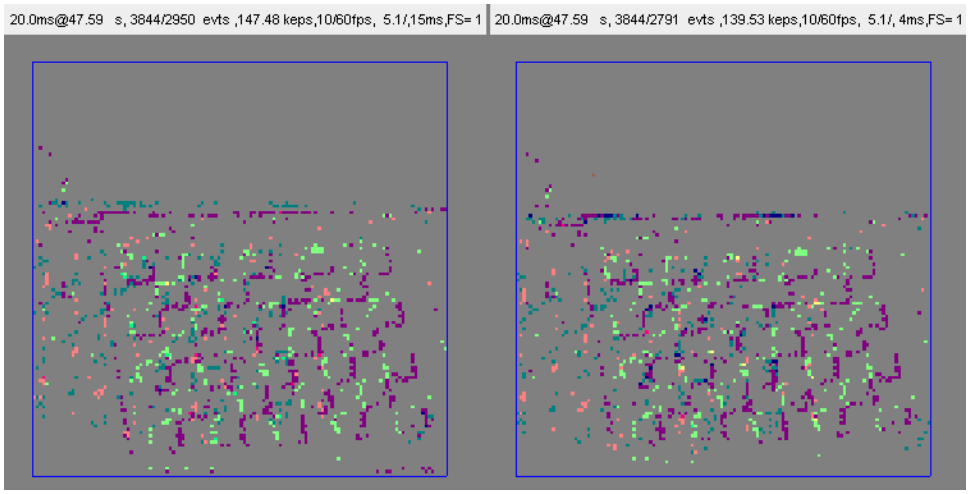

Figure 3.22: For the left camera, ON events are green and OFF events are purple while for the right camera, ON events are salmon and OFF events are cyan. The image on the left is not rectified while the right image has been rectified

3.4 Conclusion

We have successfully adapted Bouguet's calibration procedure to calibrate event-based cameras. The main modifications are the generation of frame images of calibration patterns and the creation of an epipolar correspondence map to rectify the views. The adaptation of the frame-based algorithm to event-based was conducted by simply reconstructing frames of approximated intensity levels. This method does not take advantage of the properties of the event data stream, namely the explicit spatio-temporal dynamics of the data. New event-based methods can surely be derived that use the event-based data to perform the calibration. This will be an interesting improvement with many possible advantageous outcome such as automatic and continuous calibration. The main advantage of using the frame-based approach is the availability of robust procedure that take care of all the geometric computations. Thus, using our adaptation of the frame-based Bouguet's calibration procedure, we are ready for using a calibrated setup of stereo DVS. The next chapter will address how to perform 3D scene reconstruction using this stereo vision setup.

Bibliography

[Bouguet, 2008] Bouguet, J.-Y. (2008). Calibration Toolbox for Matlab, http://www.vision.caltech.edu/bouguetj/calib_doc/index.html.

[Hartley, 1997] Hartley, R. (1997). In defense of the 8 point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593.

[Hartley and Zisserman, 2003] Hartley, R. and Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge Univ Press, second edition.

[Heikkila and Silven, 1997] Heikkila, J. and Silven, O. (1997). A four-step camera calibration procedure with implicit image correction. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1106–1112.

[Lichtsteiner et al., 2008] Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128*128 120dB 15us latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid State Circuits*, 43(2):566–576.

[Longuet-Higgins, 1981] Longuet-Higgins, C. H. (1981). A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135.

[More, 1977] More, J. J. (1977). The Levenberg-Marquardt algorithm: implementation and theory. *Lecture notes in mathematics*, 630:105–116.

[Sutherland, 1963] Sutherland, I. (1963). Sketchpad: a man-machine graphical communication system. *Proceedings of the 1963 Spring Joint Computer Conference*, (574):45–53.

[Zhang, 1999] Zhang, Z. (1999). Flexible camera calibration by viewing a plane from unknown orientations. *International Conference on Computer Vision*, 1:666–673.

4

Event-Based Stereo Vision: Correspondence Solving

Event-based computation is particularly suited to temporal computation. In chapter 2 we showed that event-based algorithms can process spatial information as well as frame-based algorithms. The exploitation of accurate temporal information can be used to solve demanding tasks where both spatial and temporal information is useful.

We successfully calibrated a stereo vision setup with two event-based cameras in chapter 3. In this current chapter we propose to use the system to perform stereo vision 3D reconstruction by exploiting the event-based data. We specifically use experimental data obtained from the Dynamic Vision Sensors. We will first review the problem of stereo vision correspondence and propose a novel event-based solution for reconstructing the depth of a scene captured by the stereo DVS.

4.1 Definitions

4.1.1 Disparity

Once the parameters of the cameras are known, the main difficulty in 3D scene reconstruction comes from finding the corresponding pair x_1 and x_2 representing the same real point X in both image planes. Thus for each pixel of the first image, we have to find the matching pixel in the second image in order to deduce the 3D position of the real points of the scene.
The relation between a pixel coordinate (x, y) and the real point imaged (X, Y, Z) is $(x, y) = (fX/Z, fY/Z)$, see figure 4.1.
Once the cameras are calibrated and rectified, their image planes are coplanar so that each epipolar line is parallel to the x-axis of the image. We can then define for each camera the equation 4.1, [Davies, 2005].

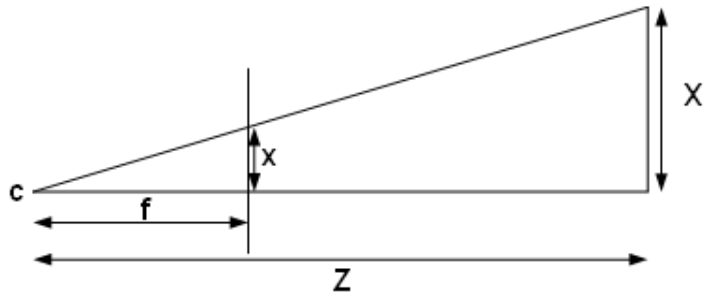

Figure 4.1: Relation between x, f and X

$$\begin{aligned} x_1 &= (X - T/2)f/Z \\ x_2 &= (X + T/2)f/Z \end{aligned} \tag{4.1}$$

where T is the distance between the two camera centers, as seen in figure 4.2.

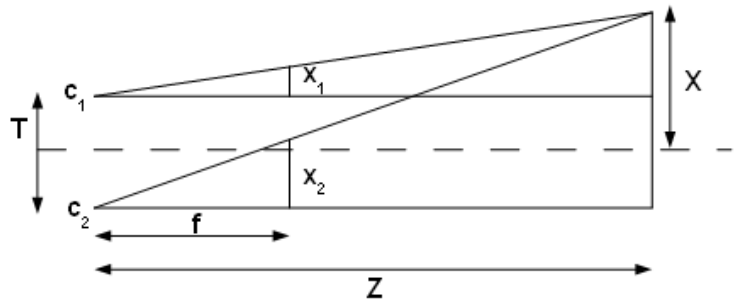

Figure 4.2: Geometrical relations between x_1, x_2 and X when the camera are rectified.

From equation 4.1 we can see that the disparity between the two pixels x_1 and x_2 is directly proportional to the depth of the real point X. This is written as in equation 4.2

$$D = x_1 - x_2 = (X - T/2)f/Z - (X + T/2)f/Z = Tf/Z \tag{4.2}$$

So that the depth Z can be computed as in equation 4.3

$$Z = Tf/(x_1 - x_2) \tag{4.3}$$

Thus the depth in the image can be computed by triangulation when the matching x_1 and x_2 points are known. Now the problem is to actually determine these correspondences based only on the imaged data.

4.1.2 Correspondence problem

The correspondence problem is thus stated as: knowing the position of an object's projection on pixel x_1 in the first image, what is its corresponding pixel x_2 in the second image?

4.1.3 Stereo-correspondence in computer vision

The stereo-vision correspondence problem has been extensively studied in computer vision for which extracting 3D structure from images has numerous applications from robotics to object recognition and industrial monitoring. Most of the solutions proposed during the past forty years directly addressed the issue as a theoretical algorithmic problem, without trying to emulate the constraints of biological systems. We review here those methods before looking at neuroscience inspired models in the next section.

All proposed solutions focus on comparing specific features in the image data frame to correctly associate the matching pixels. It is important to note that, in computer vision, an image is a frame of pixel values. These values represent the illumination intensities perceived during the time between two frames.
The main difficulty of the stereo correspondence problem is to avoid ambiguous features and results. To evaluate the performances of the matching methods, their results are often displayed in disparity maps that can be quickly compared.

The computer vision methods can be categorized into two classes: the local and the global methods [Brown et al., 2003].

Local Methods

The local methods compare the features of the pixels in a restricted local area around the target pixel. The most local computation is done on one pixel only, where intensity or color values could be used as features. Unfortunately such features are too simple and ambiguous to give robust results at the scale of one pixel only.

Windows Matching

Windows (or Block) matching study a larger area around the pixel. Combining the simple features of neighboring pixels it is possible to derive more specific signatures for these 2D windows. These signatures can be for example the mean and variance of the window's pixels intensities.

The metrics used to compare these signatures depends on the compromise between performances and computing costs. The traditional approach is to use the sum of squared differences (SSD) (see equation 4.4) which is computationally more efficient [Brown et al., 2003] than the standard statistical similarity measure using normalized cross-correlation (see equation 4.6) as pioneered by Hanna et al. [Hannah, 1974] . The sum of absolute differences (SAD) (see equation 4.5) is often preferred to SSD for its reduced computational cost [Brown et al., 2003].

$$\sum_{u,v} (I_1(u,v) - I_2(u+d,v))^2 \tag{4.4}$$

$$\sum_{u,v} |(I_1(u,v) - I_2(u+d,v)| \tag{4.5}$$

$$\frac{\sum_{u,v} (I_1(u,v) - \overline{I}_1)(I_2(u+d,v) - \overline{I}_2)}{\sqrt{\sum_{u,v} (I_1(u,v) - \overline{I}_1)^2 (I_2(u+d,v) - \overline{I}_2)^2}} \tag{4.6}$$

where $I_1(u,v)$ is the matrix window of size (u,v) for image I_1, and $\overline{I}_1$ the mean value of I_1 intensities.

Instead of selecting the best convolution result, another way to compare two windows is by defining a distance function and minimize it using gradient descent or other optimization techniques, see [Brown et al., 2003] for a review.
But overall, all windows-based methods are sensitive to ambiguities due to homogeneous areas, exposition and illumination variability as well as noise in the image [Banks and Corke, 2001]. Normalization and pre-processing of the image can reduce the effect of this limitation [Brown et al., 2003]. Rank metrics were proposed by Zabih and Woodfill [Zabih and Woodfil, 1994] that pre-process (by census-transformation, the transformation of the image into ranks of predefined characteristics) the windows before comparing the obtained rank instead of the direct features. This reduces the effect of intensity variations but also lower the amount of information available for discriminating the possible matches. The comparison of the matching windows is usually done using the Hamming distance [Hamming, 1950] which adds to the computational costs of the algorithm [Brown et al., 2003].

Feature matching

Another limitation is the effect of depth discontinuity on the similarity between corresponding windows. The differences in the corresponding windows might

prevent a match. To solve this problem and also remove the ambiguity due to homogeneous areas, it is possible to adapt the window matching to specific and more reliable features of the image than just intensities [Dhond and Aggarwal, 1989]. A popular feature is the edge of objects in the image. Edge extraction algorithms are fast and reliable and produce compact features that still correlate with the 3D structure of the scene. John F. Canny established the theory of edge detection in 1986 and implemented the Canny algorithm [Canny, 1986]. Unfortunately, the extracted edges can also be noisy, ambiguous or missing when obstructed. Corners formed by intersecting edges are also good features that reduce even more the potential ambiguities and for which many algorithms have been devised, such as the Harris detector [Harris and Stephens, 1988], all with variable efficacy see [Rockett, 2003] for performances comparisons. A recent and faster approach has been proposed by Rosten et al. in 2006 called Features from Accelerated Segment Test (FAST) [Rosten and Drummond, 2006]. FAST detects corners by testing a circle of n pixels (from 9 to 16 when trading off speed against accuracy). A supervised classifier is trained that boosts performances of corner extraction in following attempts. An upgraded version improving repeatability is presented in [Rosten et al., 2010] as FAST-ER. More generally, "interest points" are features in the image that can discriminate objects and thus serves as landmarks for stereo correspondence [Schmid et al., 2000].
More relevant features than edges and corners may exist that are difficult to know of beforehand. Automatic feature detectors solve this problem by defining the best feature from the image itself. One such algorithm is the Scale-invariant feature transform (SIFT) [Lowe, 2004]. It requires a training images set on which to build a database of feature descriptors. Interest points are located using differences of multiple scale and different standard deviations Gaussians. The differences of Gaussians is an approximation of the Laplacian of Gaussians (see [Marr, 1982], a filter that highlight rapid spatial change of intensity
[Rodieck and Stone, 1965] [Lowe, 2004]. The features are then defined as histograms of local oriented gradients around the interest point. The interest points are indexed by their location, scale and orientation for further correspondence computation. The SIFT features are invariant for scale and rotation, making them more robust than previously studied features. The SIFT algorithm involves has a high computational cost at the matching process.
Improving on SIFT in term of speed and robustness, the Speeded Up Robust Features detector (SURF) [Bay et al., 2006] follow the same principles as SIFT but implements the detection of interest points by approximating the Hessian matrix instead of using the differences of Gaussians. Interest points are here the locations where the Hessian matrix determinant is maximum. The Hessian matrix is the matrix of second derivatives of the pixel intensities with respect to their local neighborhood. It describes the local intensity gradient around a point. Another

change is that the features are not defined by an histogram of local oriented gradients but by the distribution of Haar-wavelet responses around the interest point. The Haar-wavelet responses are a set of vectors that encode and compress the information, here in a 4×4 area around the interest point. Their advantage is the added robustness from their invariance to bias in illumination [Bay et al., 2006]. SURF is a fast feature detector invariant to scale, rotation and robust to illumination differences.

Research around the extraction of local features in still ongoing. Local features detectors are a promising tool for describing and manipulating framed images components. A comprehensive review of the field are available by Li et al.
[Li and Allinson, 2008] and [Idris et al., 2009]. Using local features allows the reconstruction of sparse disparity maps, where only the depth of detected features can be computed directly, the rest of the image's depth being interpolated. This is different for the edges, as the space between edges can be reconstructed as a surface between the edges, thus giving dense disparity maps where all points of the image can have their depth directly computed and mapped, but this is still an approximation that is not always robust.
So despite the performances of interest point detectors in generating sparse disparity maps, more global methods have been studied to provide accurate dense disparity maps.

Global Methods

Instead of the local window matching, more global techniques can be used to estimate the correspondences between the images. These methods process the whole epipolar line. The most popular is using dynamic programming to map one epipolar line onto its corresponding epipolar line in the other view.

Constraints

In most situations, the structure of the real world scene is preserved in the two cameras views. This is especially true in stereo vision where the two cameras face the scene from a similar position. The structure of the scene is the relationship between its constituents (objects, light sources,...). So for example, if an object is visible to the right of another in the first view, this order will be preserved in the second view as well. This preservation of the structure can be expressed as constraints [Marr, 1982]. And some constraints can be especially useful in stereo vision. Selecting the pairs of pixels that respect some constraints may directly

solve the correspondence problem, if the scene was ideally and perfectly evaluated [Cooper, 1987]. The problem is thus to define applicable constraints that capture the structural information of the scene.

The first of these constraints if the ordering constraint. The ordering constraint defines the structure of the scene as the order of its component's edges. It states that the order of edges is the same on both corresponding epipolar lines [Arnold and Binford, 1980] [Yuille and Poggio, 1984] . The stereo correspondence problem is solved using this constraint by scanning the two epipolar lines and associating the detected edges by their order [Ohta and Kanade, 1985]. This is limited to scenes exclusively consisting of opaque surfaces. Transparency breaks the realtion between the front to back ordering of the 3D points and the lateral ordering of projected edges on the 2D view. The ordering constraint is also restricted to the ordering of edges or other invariant features. The application of this constraint creates a forbidden zone [Burt and Julesz, 1980] in 3D space around the real point of the match, where no other points can be matched without violating the constraint, see figure 4.3. The chosen order is not important as long as it preserves the scene structure [Yuille and Poggio, 1984]. But extra edges in one image can have a very detrimental effect on the matching process [Cooper, 1987]. Some algorithms are more robust to this problem, as in [Baker and Binford, 1981], and one solution is to use vertical information to distinguish edges [Cooper, 1987].

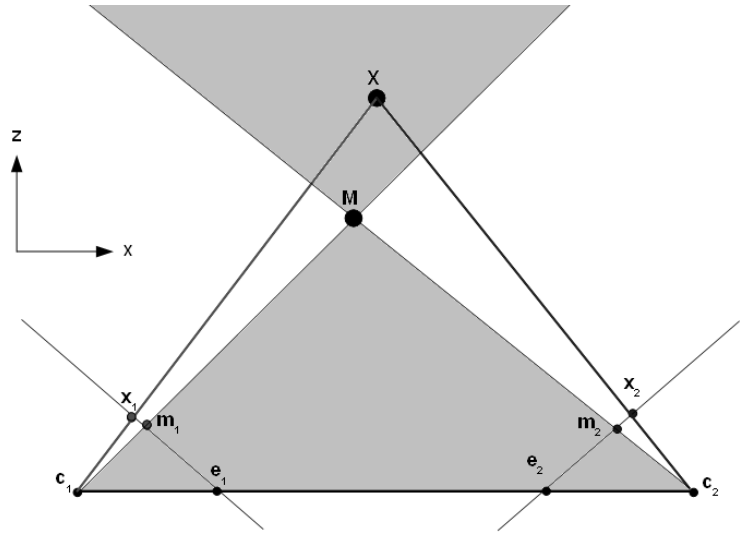

Figure 4.3: The forbidden zone of the matched pair (m_1, m_2) is highlighted in grey, point X cannot be successfully detected when using the ordering constraint, adapted from [Burt and Julesz, 1980].

The second constraint is the Uniqueness Constraint [Marr, 1974]. The correspondence between two corresponding points is bidirectional as long as there is no occlusion in one of the views. This constraint is used to detect occlusions and otherwise reduce the matching ambiguity.

A disparity limit has been observed in many psychophysical experiments, see [Qin et al., 2006] for some measurements. It has been observed that a maximum disparity range (the Panum region, see figure 4.4) [Krol and van De Grind, 1982], or gradient [Burt and Julesz, 1980], exist above which the stereopsis does not yield a single view. The disparity upper limit above which the perception of depth is erroneous has been defined as D max [Ogle, 1953] [Tyler, 1991]. Thus by defining a maximum range for the disparity it is possible to restrict the matching candidates.

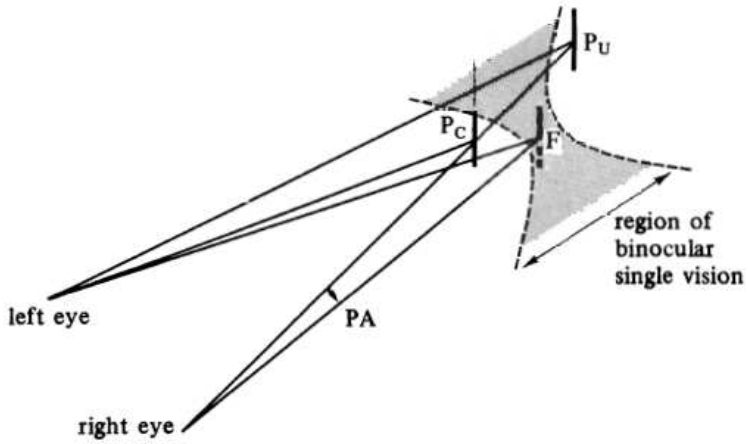

Figure 4.4: Panum's fusional area is the 3D space from which depth can be correctly estimated in human stereopsis [Krol and van De Grind, 1982].

Another constraint is the disparity continuity constraint [Marr, 1974]. Most of the image is composed of surfaces and objects that smoothly vary in depth. Abrupt changes in disparity only occur at the boundary between objects. It is thus possible to remove noise in the matching by enforcing this smoothness constraint on the matches.

The use of constraints allow to restrict the search for correspondences, but it is not very robust to noisy conditions and is often supported by additional global methods such as Dynamic Programming.

Dynamic Programming

Dynamic Programming is a recurrent search for a minimum transformation cost between two functions [Bellman, 1957]. It was first used for stereo vision in conjunction with the ordering constraint by Baker [Baker and Binford, 1981], and Ohta [Ohta and Kanade, 1985]. Dynamic Programming is here applied to the corresponding epipolar lines, to compute how to transform one into the other and thus estimate disparities between the epipolar lines. The algorithm of Dynamic Programming is attempting to minimize a cost function that measure the quality of the solution, here the distance between the two epipolar lines. The crucial part is thus to define this cost function [Amini et al., 1990]. In stereo vision many different cost function have been tried, based on pixel values
[Cox et al., 1996][Intille and Bobick, 1994] [Benosman and Devars, 1998] or features such as edges of declivity [Amini et al., 1990] [Bensrhair et al., 1996]
[Gonzalez et al., 1999].
The search for the minimum transformation takes place in a 2D square matrix (see figure 4.5) where each node N_{ij} is the cost of matching the i features of the first epipolar line with the j features of the other.

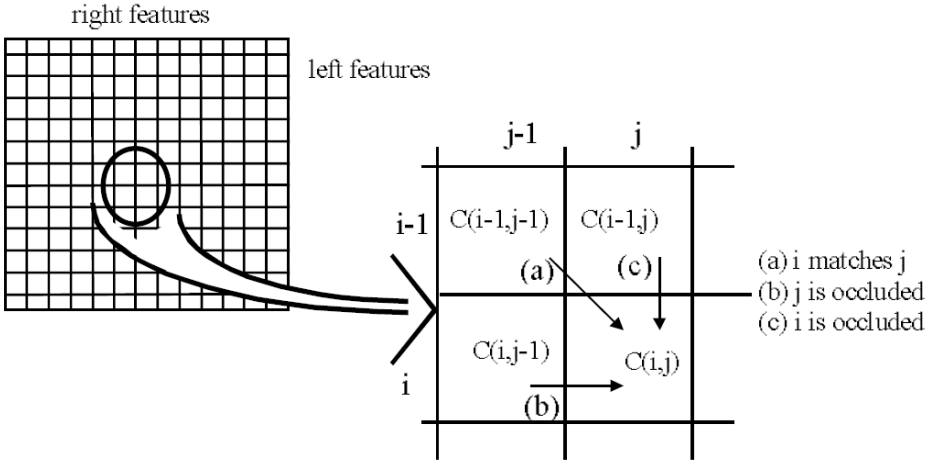

Figure 4.5: Dynamic Programming search matrix, each new row is a more refined estimation of the transformation between the epipolar lines [Gonzalez et al., 1999]

The algorithm is thus described by equation 4.7 [Amini et al., 1990] .

$$GS(x_1, x_2) = LS(x_1, x_2) + \max_{i,j \in R} GS(i, j) \tag{4.7}$$

where $GS(x_1, x_2)$ is the global score of the association between pixels x_1 and x_2, $LS(x_1, x_2)$ is the local score established comparing local features of pixels x_1 and

x_2, and $\max_{i,j \in R} GS(i,j)$ is the maximum of global scores prior to x_1, x_2 in the region R defined as in equation 4.8.

$$R = \{(x_1 - 1, x_2), (x_1 - 1, x_2 - 1), (x_1, x_2 - 2)\} \tag{4.8}$$

The matrix N_{ij} is then filled with values and the path of lowest cost can be computed that correspond to the transformation between the two lines and thus the disparities.
More global methods exists, such as Bayesian stereo matching [Barnard, 1989] [Stewart et al., 1996] and [Cheng and Caelli, 2007], or relaxation of matching probabilities [Barnard and Thompson, 1980], that complement the previously described methods.

Limitations and difficulties of computer vision stereo algorithms

According to a constantly updated benchmark by Scharstein et al.
[Scharstein and Szeliski, 2002] [Scharstein and Blasiak, 2010], the best algorithm is, in 2009, the "Segment-Based Stereo Matching Using Belief Propagation and a Self-Adapting Dissimilarity Measure" proposed by Klaus in [Klaus et al., 2006]. Its achieves high accuracy results by fitting candidates disparity planes after a more classical estimation of the disparity for each pixel using a window-based matching, supported by the detection of homogeneous regions. This mix of local and global methods allows for high performances although the use of disparity planes is a specialization for the benchmark task that may limit the use of the algorithm in some applications. So despite all the strong performances shown by recent algorithms, computer stereo vision is still difficult to achieve in noisy or complex conditions. Biological systems still prove much more reliable and efficient. The computer vision methods all process the images as frames, while the biological systems address it as a dynamic stream of visual data. We review the neuroscience stereo vision models in the following section.

4.1.4 Stereo-correspondence in neuroscience

Stereopsis, the ability to perceive depth by processing the combined input of two eyes, has been extensively studied in neuroscience. It is now clearly established that neurons of the early visual pathway are already sensitive to depth. The earliest disparity sensitive neurons are found in the visual area 1 or the cortex (V1) [Hubel and Wiesel, 1962], but other have also been detected in V2 and V3 [Poggio and Fischer, 1977] and it is likely the depth is processed more completely by including non-binocular clues (such as perspective) in later cortical areas such as the temporal cortex [Read, 2005] [Parker, 2009]. Nevertheless Julesz showed

that low level information is enough to perceive depth, without additional clues but binocular disparities [Julesz, 1971]. He designed and used random dot stereograms, now widely used as stimuli in stereopsis experiments. Dot stereograms are pairs of artificial images made of dots , with some points shifted in the second image so that seeing one in each eye creates a sensation of depth. Thus it is possible to learn about stereopsis by studying the earlier disparity-tuned neurons of the visual pathway. The visual information is first encoded in spikes by the retina ganglion cells and transmitted to the Lateral Geniculate Nucleus (LGN) before reaching the visual cortex.
Following the central visual pathway, the information from both eyes is reunited only starting from V1 [Hubel, 1959], the earliest visual cortical area. Another visual pathway targets the superior colliculus in the brainstem also seem to play a role in stereopsis [Bacon et al., 1998]. Disparity selective cells are found in the superficial layer of the superior colliculus, exhibiting a much coarser sensitivity than cortical disparity-tuned neurons [Bacon et al., 1998].
The models of stereopsis derived from neuroscience are based on the properties of these cells and their receptive fields. For the neurons to compute disparity between the left and right eyes input, data from both input must be combined at some point. Marr and Poggio proposed an algorithm compatible with psychophysical and neurophysiological data in 1979 [Marr and Poggio, 1979]. It made use of bar masks of different size and orientation to detect edges that are then matched on limited disparity range. This showed the importance of oriented edge filtering.

Thus it is important to characterize the receptive field of visual neurons. The receptive field is defined in the visual retinal space, with a position and a shape. It has positive (ON) and negative (OFF) regions according to the effect (excitatory of inhibitory) a stimuli has on the cell when the stimuli is in these regions. The complexity of receptive fields in V1 results from the connectivity pathway between the retinal cells and the V1 cells [Hubel and Wiesel, 1962]. There are two main categories of neurons in the early visual cortical area V1, distinguished by their receptive fields. Those are the simple cells and the complex cells. The simple cells have oriented center surround Gabor like receptive fields, see b) in figure 4.6 while the retinal ganglion cells exhibits simpler ON-OFF surround receptive fields (a) in figure 4.6) [Jones and Palmer, 1987]. It is important to note that the amount of excitatory and inhibitory region is balanced.

Disparity tuned cells have been classified into four different categories by Poggio and Fischer [Poggio and Fischer, 1977]. They distinguished - tuned excitatory disparity neurons, which respond strongly to binocular stimuli with small horizontal disparities, each neuron having its own preferred disparity, - tuned in-

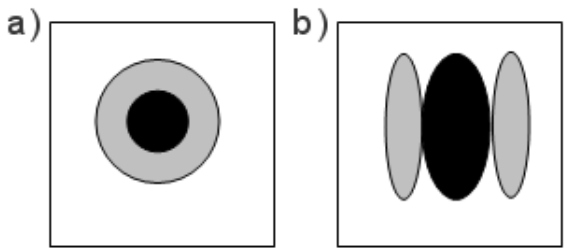

Figure 4.6: a) ON-OFF surround receptive field(RF) of retina ganglion cells. b) Oriented Gabor like RF of Simple cells in V1. Black represents excitatory response while grey represents inhibitory response.

hibitory neurons, that show maximal inhibition for small horizontal disparities and maximal excitation at large disparities, - near neurons, which are sensitive to stimuli in front of the fixation plane (crossed disparities) and - far neurons, that show opposite of the near neurons response. Tuned excitatory cells can also be decomposed into three diferent response beahviour: tuned near with a preference for crossed disparities, tuned far, and tuned zero responding strongly for very small disparity.

Currently, two hypothesis co-exist to explain how receptive fields of different eyes can be used to compute disparity: the position-shift and the phase-shift theories. The position shift theory assumes that the receptive fields of the simple cells in each retina occupies the same set of locations and that disparities arise from activity detected at different positions in the two retina. So a non-zero disparity preference would emerge for having two receptive fields in the exact same position in both retina. Evidence of such mechanisms have been observed in the cat [Hubel and Wiesel, 1962] and the owl [Pettigrew and Konishi, 1976] . The phase-shift theory does not requires receptive fields to cover the same locations but instead define a preference for zero disparity by having Gabor filters overlapping. Receptive fields of different phase can then overlap when neighboring each others. Ohzawa et al computed that an optimal phase-shift would be 90 degrees and derived a model of a complex cell able to estimate disparity. This model is called the disparity energy model [Ohzawa, 1998], see figure 4.7.

Many more details on stereopsis is reviewed by Gonzalez in
[Gonzalez and Perez, 1998]. These models differ from computer vision algorithms by using many local computational units in parallel, dealing with a dynamic stream of visual data, in a way which is difficult to implement in either software or hardware yet. But recent progress in neuromorphic engineering attempt to emulate some aspects of stereo vision, as we will see in the next section.

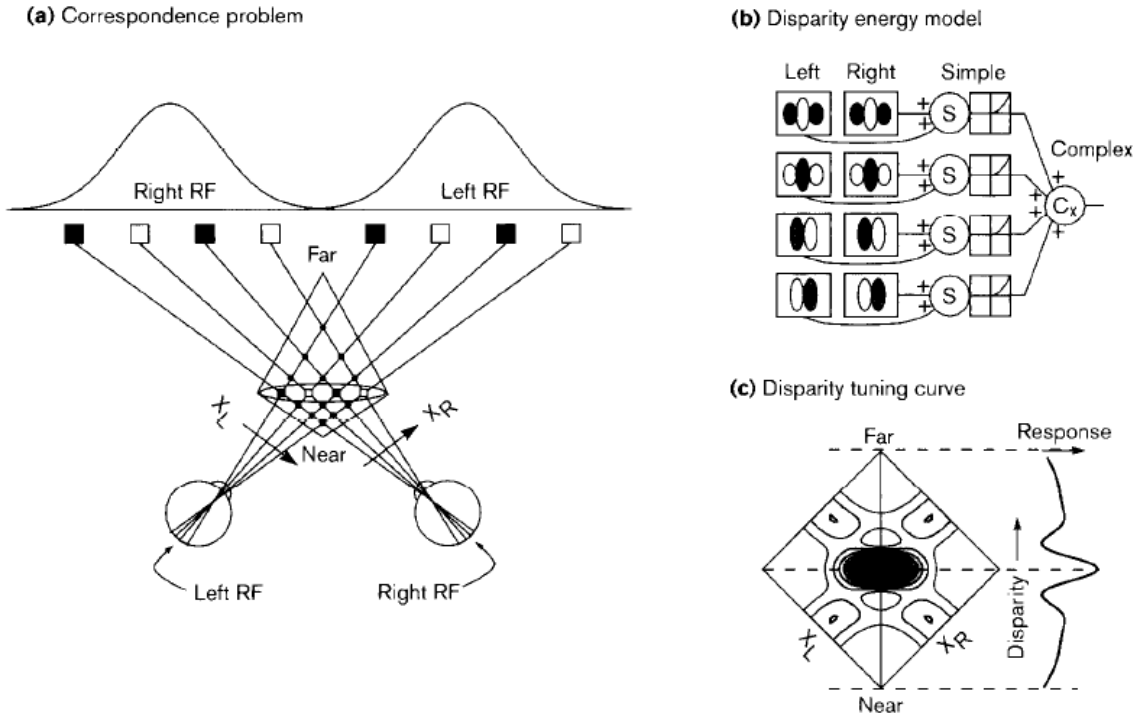

Figure 4.7: a) reconstruction of the depth from combination of the left and right receptive field (RF) of complex cells is subject to false matches. On top the response curve of the RFs. b) The complex cells RF as a combination of simple cells gabor filters. c) Predicted response for a binocular RF, from wich the disaprity curve (on the right) can be derived. [Ohzawa, 1998]

4.1.5 Stereo-correspondence in neuromorphic engineering

Mahowald et al. [Mahowald and Delbrück, 1989] implemented cooperative stereo vision in a neuromorphic chip in 1989. The stereo vision sensors implemented was composed of two 1D pixel arrays of 40 neuromorphic pixels each, connected to a correlator array of 3×40 elements, for which a simplified illustration is shown in figure 4.8. The use of local inhibition driven along the line of sight implement the uniqueness constraint (one pixel from the one view is associated to only one in the other, except during occlusions), while the lateral excitatory connectivity gives more weight to co-planar solutions to discriminate false matches from correct ones. In 1992, Mahowald presented an improved algorithm that adds a layer of filters tuned to different spatial frequencies of multiple scales. The response of these filters is positively fed back to the correlator's units to improve the detection of correct matches [Mahowald, 1992]. These methods require a great amount of correlator's units to deal with larger visual sensors resolution.

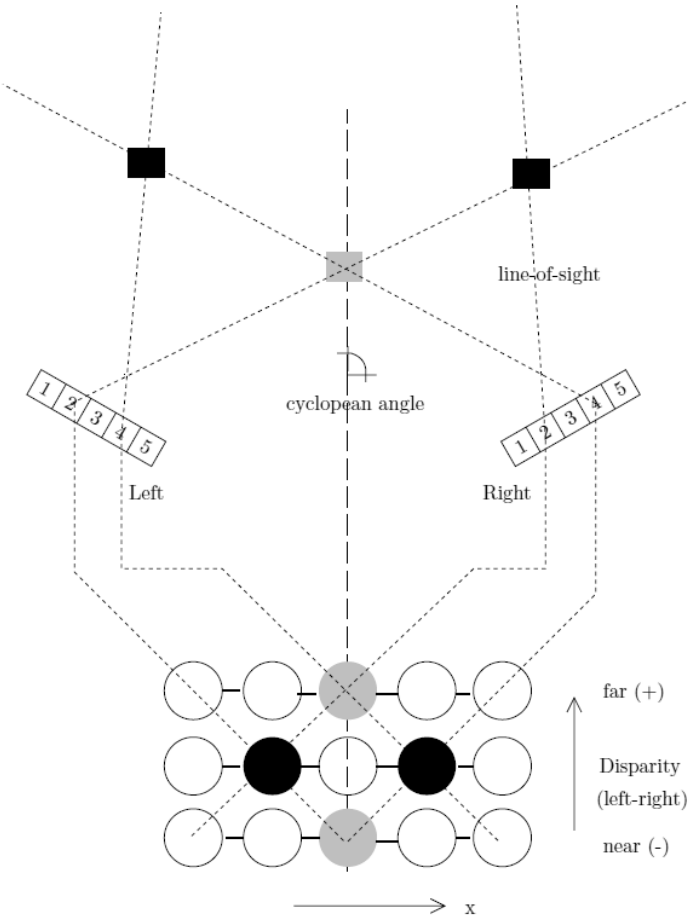

Figure 4.8: Neuromorphic stereo vision designed by Misha Mahowald. The cooperatiove stereo matching is performed by a correlator array whose elements are linked by inhibitory (dashed lines) and excitatory (plain line) connections. The correct matches (black circles) strengthen each other while inhibiting the false matches (grey circles). [Mahowald, 1992]

Kogler and colleagues [Kogler et al., 2009] have described a frame-based usage of the event-based DVS cameras in 2009. They designed an event to frame converter to reconstruct frames and then tested two stereo frame-based vision algorithms, one window-based and one feature-based using center-segment features, see [Shi and Tomasi, 1994].

Delbrück has implemented a real event-based stereo tracker that tracks the position of a moving object in both views using an event-based cluster tracker and then reconstruct the position of the object in 3D [Delbruck, 2009]. This efficient

and fast method lacks resolution on smaller features and is sensitive to noise when too many objects are present.
Eric Tsang and Bert Shi implemented in [Tsang and Shi, 2004] both position and phase tuned model of the disparity energy model [Ohzawa, 1998] on multiple neuromorphic chips and using the sustained ON and OFF response of two event-based cameras. They designed gabor-type filter chip to emulate the simple cells receptive fields. The system was limited to the computation of only one tuned disparity at a time. In 2008, Shimonomura, Kushima and Yagi implemented disparity energy model with multiple disparity tuning to perform stereo vision with two silicon retinas
[Shimonomura et al., 2008]. They simulated elongated receptive fields to extract the disparity of the scene and control the vergence of the cameras. The approach is frame-based and allow to extract coarse disparity measurements to track object in 3D.

4.2 Stereo vision using event-based data

As we have seen in the previous sections, many solutions have been proposed, combining many approaches inspired from biology and statistics. Their current limitations could be in part related to the frame data format used in computer vision. In this section, we propose an event-based algorithm performing stereo vision matching on dynamic spatio-temporal data and exploiting the advantage of the asynchronous temporal resolution of event-based data.

4.2.1 Event-based matching

A point moving in real space triggers a change of luminance in the field of view of the cameras as denoted by $X(t)$. In the case of a stereo camera setup, $X(t)$ will project onto the n^{th} camera's focal plane $\mathcal{C}_n$ according to the relationship $p_j^n(t) = P^n X(t)$ where P^n is the projection matrix associated to the camera, see figure 4.9. At this exact time t, the change of luminance will affect pixels and eventually generates events for both cameras. The timing of these events will not correspond to the exact timing of the real space luminance change. Following the temporal properties of the asynchronous event-based cameras, in an ideal case, the set of corresponding $p_j^n(t)$ should be recorded with equal time values as they are the consequence of the same event occurring at a fixed time. We could then expect to directly match the events based on their exact incoming times.
Unfortunately, due to the finite spatial sampling resolution, the pixel aperture and electronic architecture constraints, events do not usually arrive at the same time. Following its biological role model, the DVS pixel is non-ideal in some respects: it has non-zero and variable latency from visual stimulus to event generation, random jitter in the timing of events and it also generates events independently of the visual scene stimuli due to noise and leakage in the electronic processes. Thus it is impossible to associate matching pixels based on their exact timing.
The situation is further complicated by the activity of other pixels in the arrays due to the camera's operating and readout principles. A moving scene point $X(t)$ will generate corresponding events on pixels p_i^1 at time t_1 and p_j^2 at time t_2. As shown in figure 4.9, when pixel p_i^1 generates an event at time t_1, other pixels (for example p_k^2 or p_l^2) that have no a priori relationship with $X(t)$ can also generate events in close temporal proximity.
This can be due to other scene-related stimulations or background noise activity. In a typical staring application environment, where only a few moving objects need to be matched, most of these effects can be neglected and only latency and jitter are of concern. The latency of the pixel response is the time between the

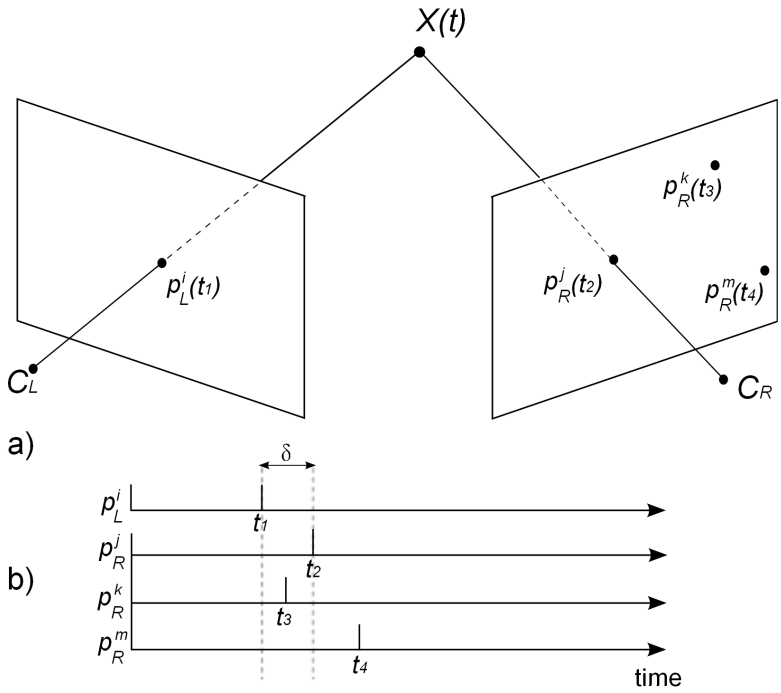

Figure 4.9: Jitter between corresponding events: two events generated by the real point X at time t will be detected by corresponding pixels p_i^L and p_j^R at slightly different times t_1 and t_2 (the order between left and right events is not fixed). This time difference δ, the jitter, is due primarily to the functioning principles of the pixels. During this time difference δ, other pixels such as p_k^R may become active so exact timing does not allow to directly detect stereo matches.

occurrence of an illumination change at the photoreceptor and the corresponding event output. This variable delay time is, at typical illumination conditions, limited by the photoreceptor bandwidth which is proportional to photocurrent and hence to pixel illumination. For a given scene, the event latency can be assumed to be of the same order of magnitude and constant for a pair of pixels being stimulated at comparable illumination and by similar temporal contrasts.
The jitter is defined as being the standard deviation of latencies of one pixel stimulated by identical temporal contrast changes. The jitter is due to photocurrent and dark current shot noise in the photoreceptor and readout noise in the pixel circuitry. It can be modeled as Gaussian distributed around a mean latency. The mean latency is dependent on the scene's background illumination, and is constant but different from pixel to pixel (fixed-pattern-noise) and from chip to chip. This jitter, apart from effects on latency due to different illuminance of pixels, is the dominant source of variations in the pixel event timings. For a typical indoor application illumination range, the illumination range runs from 10 to 300

lux and the jitter of one pixel is of the order of 30 to 200 μs [Lichtsteiner, 2006] [Lichtsteiner et al., 2008].

A pixel p_2 with mean latency of t_2 will in most cases deliver its event after pixel p_1. The difference $\delta = t_2 - t_1$ of mean latencies of each given pair of pixels across two large arrays of pixels is zero. However, for establishing the size of a search window for stereo event matching, a mean upper bound to the difference of mean latencies has to be taken into account. An optimum search window size can be established theoretically only if this parameter is known. The mean latency has been measured between 15μs and 4ms depending on the illuminance of the chip [Lichtsteiner et al., 2008]. Following these observations time search windows sizes between 500 μs and 2ms were used in this chapter's experiments.

Time window

With two cameras denoted L and R, if a change in a point of the scene $X(t)$ produces events E^L and E^R activating pixels $p_j^L(t) = P^L X(t)$ and $p_k^R(t) = P^R X(t)$ with $p_j^L(t) = \begin{bmatrix} x_j & y_j & t_j \end{bmatrix}^t$ and $p_k^R(t) = \begin{bmatrix} x_k & y_k & t_k \end{bmatrix}^t$, we define $A_{k,j}$ as the set of all the temporal activations of pixel $p_k^R \in \mathcal{C}_R$ (i.e. belonging to the right camera) happening within a time window δ from those of pixel $p_j^L \in \mathcal{C}_L$:

$$\begin{aligned} A_{k,j} = \{t_k \in [t_j - \delta/2, t_j + \delta/2] | E^L(x_k, y_k, t_k) \neq 0 \\ \text{if } E^R(x_j, y_j, t_j) \neq 0\}. \end{aligned} \tag{4.9}$$

Using the definition of $A_{k,j}$ given by equation(4.9), we can then write the conditional probability:

$$P(A_{k,j}) = P(E^L(x_k, y_k, t_k) \mid E^R(x_j, y_j, t_j)) \tag{4.10}$$

for $t_k \in [t_j - \delta/2, t_j + \delta/2]$.
This gives us:

$$P(A_{k,j}) = \frac{P(E^R(x_k, y_k, t_k) \cap Ev^L(x_j, y_j, t_j))}{P(Ev^L(x_j, y_j, t_j))} \tag{4.11}$$

We can then define a probability field b^j such that :

$$\begin{array}{rcl} b^j : \mathbb{N}^2 & \rightarrow & [0, 1] \\ (x_k, y_k) & \mapsto & b^j(x_k, y_k) = P(A_{k,j}) \end{array} \tag{4.12}$$

If B^j is the matrix of elements b^j, then it contains the probability of all temporal appearance of pixels $\in \mathcal{C}_R$ for the pixel $p_j^L \in \mathcal{C}_L$:

$$B^j = \begin{bmatrix} P(A_{1,j}) & P(A_{2,j}) & \cdots & \cdots \\ \vdots & \ddots & \ddots & \vdots \\ \cdots & \cdots & \cdots & P(A_{k,N}) \end{bmatrix} \tag{4.13}$$

with N being the total number of pixel of the camera. According to noise parameters and the period of observation δ, most of $A_{k,j} = 0$, i.e. B^j, is a sparse matrix which can be seen as a coarse expression of the corresponding epipolar line of pixel $m_j^1 \in \mathcal{C}_2$.

The probability of pixels activation is directly linked to their possibility of match. Let $e^L(t)$ be an event from the set E^L and $R(e^L(t))$ the set of events in the right camera occuring in a time window around the time t of the left camera event $e^L(t)$. When $R(e^L(t))$ is observed on short periods of time, the values of the corresponding right camera pixel to $e^L(t)$ can be discriminated by its distance to the corresponding epipolar line, see figure 4.10.

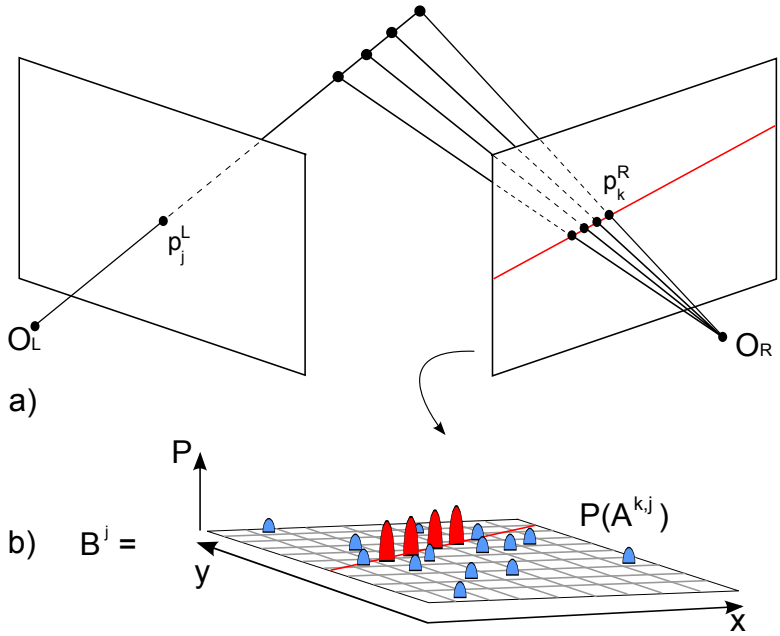

Figure 4.10: Epipolar geometry structure from the probabilistic co-activity of events. a) Matching events to p_j^L fall closest to corresponding epipolar line l. b) This is reflected in the probability field B^j that for a left pixel p_j^L describes the co-activation probability in the right pixel array.

To exploit this probability of co-activation, we define within a time window δ, that for an incoming event $e_1^L(t)$ in the left view, exists a set $B^j(e_1^L(t))$ of right view events e_k^R of respective times t_k as defined by equation 4.14.

$$R(e^L(t)) = \left\{e_k^R\right\} \text{ for all } k \text{ where } \|t_k - t\| < \delta \tag{4.14}$$

where t_k are the incoming times of events e_k^R and t the time if event e_1^L, and δ is the time window's size.

Distance to the epipolar line

The matching approach we choose here is to filter out as much as possible the wrong matches so that only the true match remains. The criteria to rule out false matches will be applied to events within the time window δ. Because we can still apply a selection on the shortest time difference, the remaining false matches that pose problems lie in the time interval between the two truly matching events. To discriminate these, we compute the matching decision between each incoming e_i^L events and its associated set of right events $R(e_i^L)$ by using the distance between the activated right pixels p^R generating the events $e^R \in R(e_i^L)$ and the right epipolar line l_i corresponding to the left pixel p_i^L, see figure 4.11. Following that the projection of a real world point X is p_i^L in the left image plane, the back projection from x to X is a line L in the world 3D space. This line projects itself onto the second image plane as l_i. l_i is called the epilolar line and represents all possible p^R projections of X in the right image. The corresponding point p_j^R of p_i^L can only lie on the epipolar line l_j.
We then compute the matching decision between each incoming e_j^L events and its associated set of right events $R(e_j^L)$ based on the distance between the right events $e_i^R \in R(e_j^L)$ and the right epipolar line l_j corresponding to the left event e_j^L. From now on we refer to the i^{th} right event $e_i^R \in R(e_j^L)$ as simply e_i^R. The calibration of the stereo setup gives us the fundamental matrix F as explained in chapter 3. The epipolar line l_j is derived from the fundamental matrix F by equation 4.15.

$$l_j = F^T p_j^L \tag{4.15}$$

and we use the euclidean distance 4.16 to derive and then sort the set of closest points S as defined by 4.17.

$$d(e_i^R, l_j) = \frac{|ax_i + by_i + c|}{\sqrt{a^2 + b^2}} \tag{4.16}$$

where (x_i, y_i) are the coordinates of event e_i^R in the camera view, and l_j is defined by the equation $ax + bx + c = 0$.

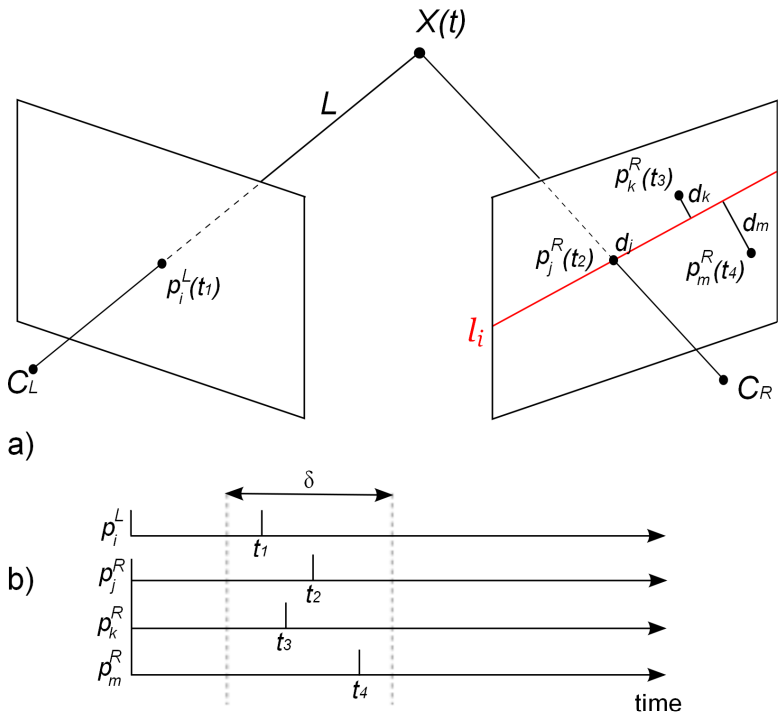

Figure 4.11: The match between corresponding events is based on the distance of the match candidates toward the epipolar line corresponding to the left event. a) Two events generated by the real point X at time t will be detected by corresponding pixels p_i^L and p_j^R at slightly different times t_1 and t_2. d_j, d_k and d_m respectively measure the distances between the epipolar l_i and the pixels p_j^R, p_k^R and p_m^R b) During this time window δ, other pixels, such as p_k^R and p_m^R, may become active, but only the closest pixel to the corresponding epipolar line l_i is likely to be the correct match.

$$S(e_j^L(t)) = \{e_i^R\} \text{ for all } i \text{ and } e_i^R \in R \text{ where } d(e_i^R, l_j) < \theta \tag{4.17}$$

where θ is the maximum distance allowed, in pixels.

Additional constraints

To remove remaining ambiguities we add the following constraint in order to select the best match for e_j^L within S.

- The polarity of the events, ON and OFF, is taken into account so that ON events are only matched to ON events, and similarly for OFF events.

- The uniqueness constraint is applied so that an event cannot be matched twice.
- The past temporal activity, see further in equation 4.19.
- The ordering constraint, see further in equation 4.20.

Past temporal activity

The past temporal activity T, defined in equation 4.18 is further used to select a single match as defined in equation 4.19.

$$T(e(x,y)_t) = \sum_i sign(e(x,y)_{t_i}) \tag{4.18}$$

where $e(x,y)_{t_i}$ are events occurring at camera coordinates (x,y) for times $t_i <= t$. T is the sum of past event activities for pixel (x,y).

$$m(eL_j)(t) = \min(|T(e(x,y)_t) - T(e(x,y)_t)|) \tag{4.19}$$

Ordering constraint

The ordering constraint is applied to the matching process to enforce a left-right order of matching occurrence. There is no need for edges extraction as the incoming events represent temporal contrast. In effect, each event thus correspond to a moving edge when the cameras are kept stationary. The constraints is implemented by the equation 4.20

$$m(x_1) = x_2 \text{ if } x_2 > x_1 \tag{4.20}$$

where $m(x_1)$ is the right match for the left point x_1 and x_2 a point in the right view.

When a corresponding event is found, a binocular disparity events is generated that additionally encodes the computed disparity.

Implementation

We implemented our stereo matching algorithm as a filter named StereoOnFundamentalMatrix4 in the Java Address Event Representation (jAER) framework developed by Tobi Delbrück [Delbruck, 2009].

The pseudo-code algorithm for this filter can is as follows:

```
/// array of possible right event candidates for a given left event
Event rcandidates[]
for each left event eL
  // fidn possible candidates
  for each left event eR
    rcandidates[i] = eR if abs(eR.time-eL.time)<timeWindowSize;
    i++;
  end for
  epipolarLine = computeEpipolarLine(e.x,e.y,FundamentalMatrix);
  Event eRMatch = closestEventToEpipolarLine(rcandidates,epipolarLine);
  if(eRMatch.polarity==eL.polarity and eRMatch.disparity not set)
    eRMatch.disparity = eR - eL;
    generate new disparity event
  end if
end for
```

The result of the matching computation is always expressed as a disparity event carrying information on time, x,y location and disparity computed for the left camera. These events can be further processed by additional filters to display disparity maps or 3D reconstructions.

Setup

We use in our experiment the same stereo DVS setup as calibrated in the previous chapter 3. This consist of two DVS of 128 by 128 asynchronous pixels [Lichtsteiner et al., 2008] mounted in stereo. Their clocks are synchronized by a dedicated board developed by Patrick Lichtsteiner. The lenses are 10 to 30 mm focal length, here set at 10mm. The calibration is performed as described in chapter 3. The aperture and the focus of the lenses is set manually while checking the results by vibrating the DVS so that the features of the scene are observable.

4.2.2 Results

Disparity maps for a moving pen

We present here the results of our event-based stereo matching algorithm. Disparity maps are created from the disparity events generated by the stereo matching filter. The computed disparity is color-coded from blue to red for result respectively varying between 0 and 128 pixels. The default value for background unprocessed pixels is 0.

We recorded a scene in which a pen is moved sideways at three different depths, as illustrated in figure 4.12. The obtained disparity maps are shown in figure 4.13 with parameters set as follows: 2ms time window, maximum distance to epipolar line, 3 pixels, maximum difference in temporal activity summation, 3 events.

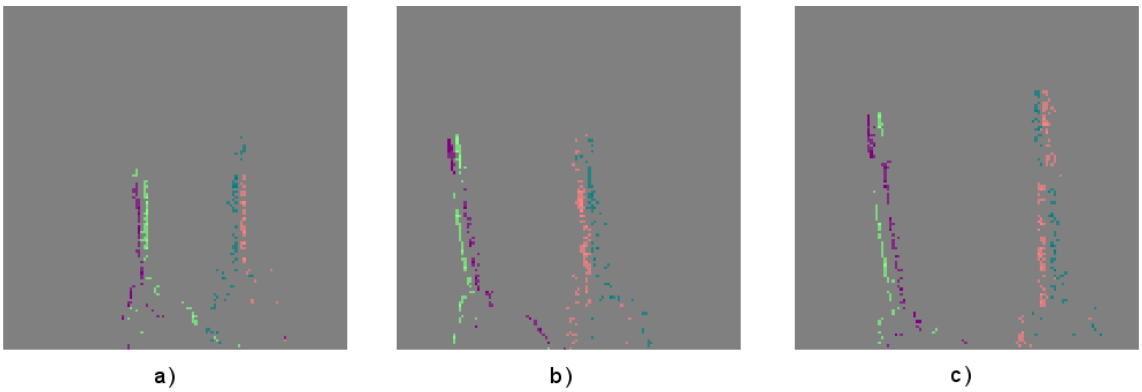

Figure 4.12: Moving pen recorded at three different depths, a) far, b) middle, c) far. The left and right views are merged into one, color coded as green/violet for left ON and OFF events, and salmon/cyan right ON and OFF events.

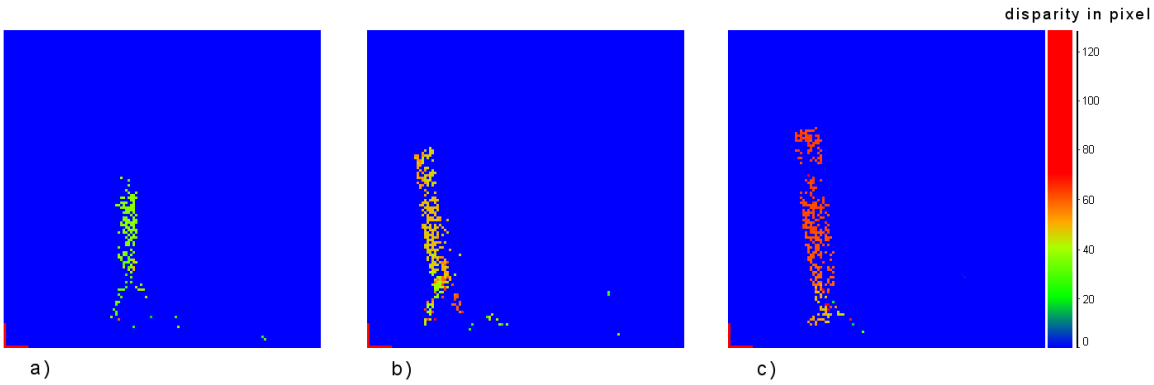

Figure 4.13: Disparity maps for pen at three different depths, a) far, b) middle, c) far. In this case red means closer to the camera and blue farther away. The estimated disparities are coherent with the actual depth of the pen.

As shown in figure 4.13, the disparity change is correctly evaluated. The pen in motion shows higher disparities when it is close than when it is far. We plot the histogram of disparities along the y axis in figure 4.14.
The disparity is measured in pixels and its distribution is well centered on the correct value. The pen is a thin object generating few events on the same epipolar lines.

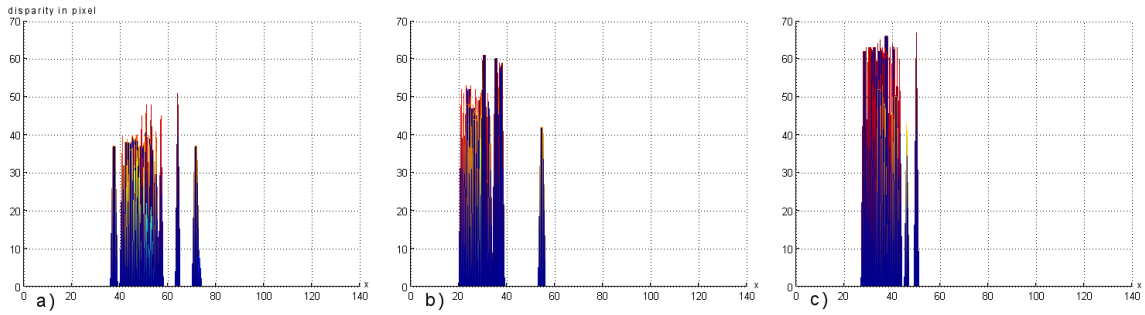

Figure 4.14: Disparity histogram on x axis for pen at three different depths, a) far, b) middle, c) far. The disparities, measured in pixels, are well centered on the correct values. The fact that the histograms are not slanted is a good indication that the matching result is not an artifact of the ordering constraint.

Disparity maps for waving hand

We performed the same experiment on a recording of a wider object: a waving hand. Three different depths are studied as shown in figure 4.15. The obtained disparity for each depth are plotted in a color-coded maps 4.16

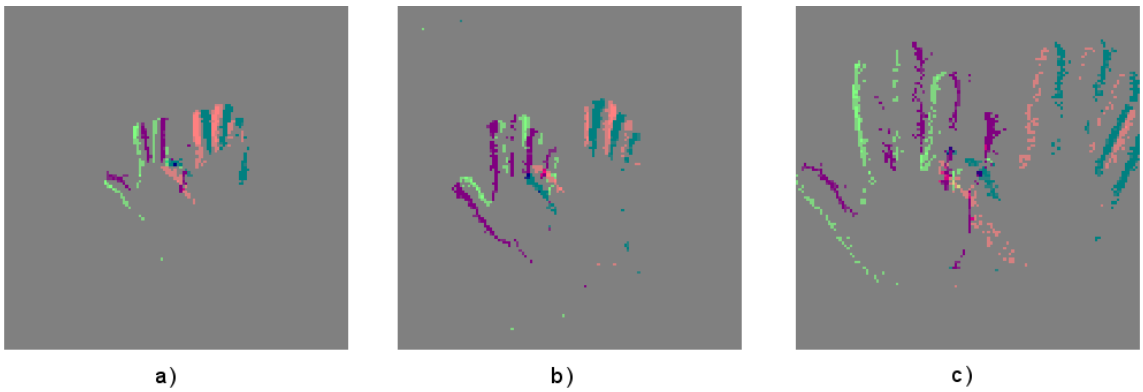

Figure 4.15: Waving hand recorded at three different depths, a) far, b) middle, c) far, color coded as green/violet for left ON and OFF events, and salmon/cyan right ON and OFF events.

The disparity are correctly evaluated although noise is present. This noise is principally due to motion and differences in both cameras sensitivity to contrast and differences in background viewed by each camera.
The distribution of disparities along the x axis is given in figure 4.17. It shows that the events are not matched randomly across the width of the hand, but that they give a correct estimation of the hand's depth.

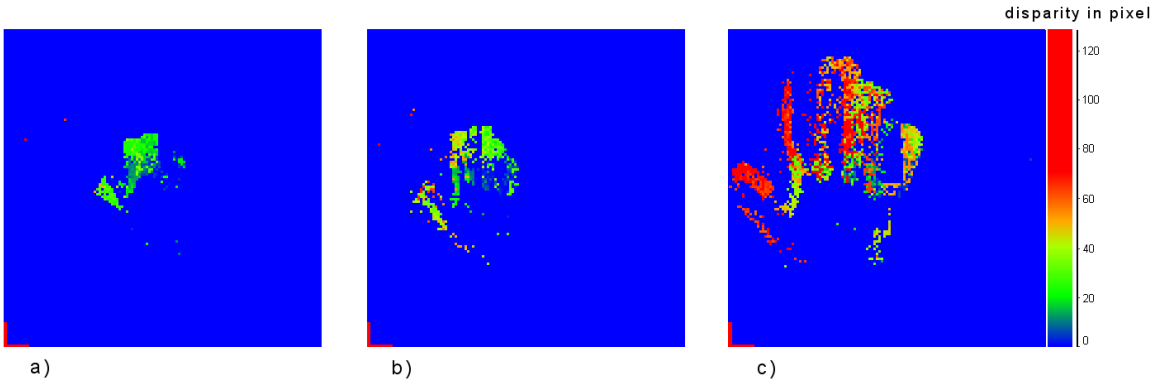

Figure 4.16: Disparity maps for hand at three different depths, a) far, b) middle, c) far. Here the results are more noisy, due to the increased inherent ambiguity as the object is wider, and also due to the accumulation time used to reconstruct the maps (20ms).

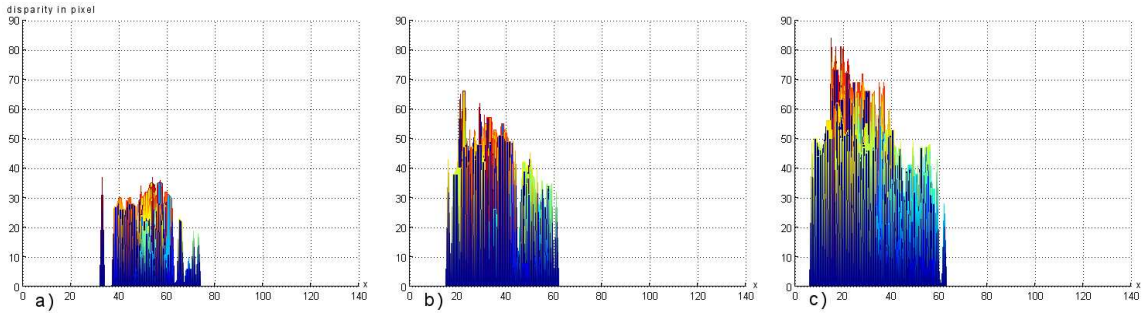

Figure 4.17: Disparity histogram on x axis for hand at three different depths, a) far, b) middle, c) far. The disparity is shown in pixels.

Disparity maps for two moving pens

The temporal dynamics also improves the discrimination between different objects. We recorded two pens moving simultaneously in the field of view of the stereo DVS setup. The result is visible in figure 4.18. The left part (a) shows the obtained disparity map, the right part shows the raw data recording of the two pens (b).

3D reconstruction of a moving pen

The output of the filter are events carrying information on the computed disparity. We can feed this events into an additional 3D display filter to observe the scene reconstruction. The 3D display filter reconstruct the 3D position of events

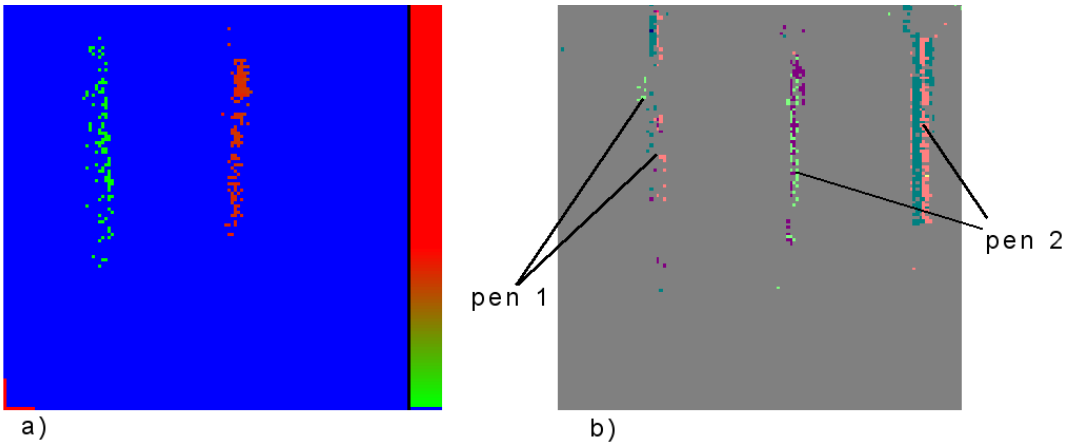

Figure 4.18: a) Disparity map and b) raw data of two moving pens at different depths. The merged view is color-coded with green/violet for left ON and OFF events, and salmon/cyan right ON and OFF events.

by triangulating the cameras coordinates and the matching pixels.

We recorded a moving pen at different depths. Figure 4.19 shows the data for four different depths illustrated in three parts. The left part (a) is the 3D reconstruction viewed from the side, so that the Z axis is horizontal (as shown in the top part where the z axis is plotted in red, the y axis is yellow and the x axis is green). The 3D reconstructed points are plotted in green when the event's polarity is positive, blue otherwise. The middle part (b) shows the obtained color-coded disparity map. The right part (c) shows the raw recorded data of the moving pen, color coded for left and right events merged into a single view. The pen's handle is transparent and sometimes only the pen's tip is apparent. The different position are correctly mapped in the 3D space as 3D events that can be further used for 3D data processing.

3D reconstruction and trajectory of a swinging ball

By performing stereo vision using the high temporal resolution of the data we are able to reconstruct fast 3D trajectories of an object. We tested this with a swinging ball, see picture 4.20. The ball is attached by a string to a point on top, from which a swinging movement is created sideways and slight forward. The 3D reconstruction of the moving ball is shown in figure 4.21 where its position is obtained by computing the center of gravity of the reconstructed points and plotted it as a violet cube. The successive position of the ball gives us a reconstructed trajectory shown in figure 4.22.

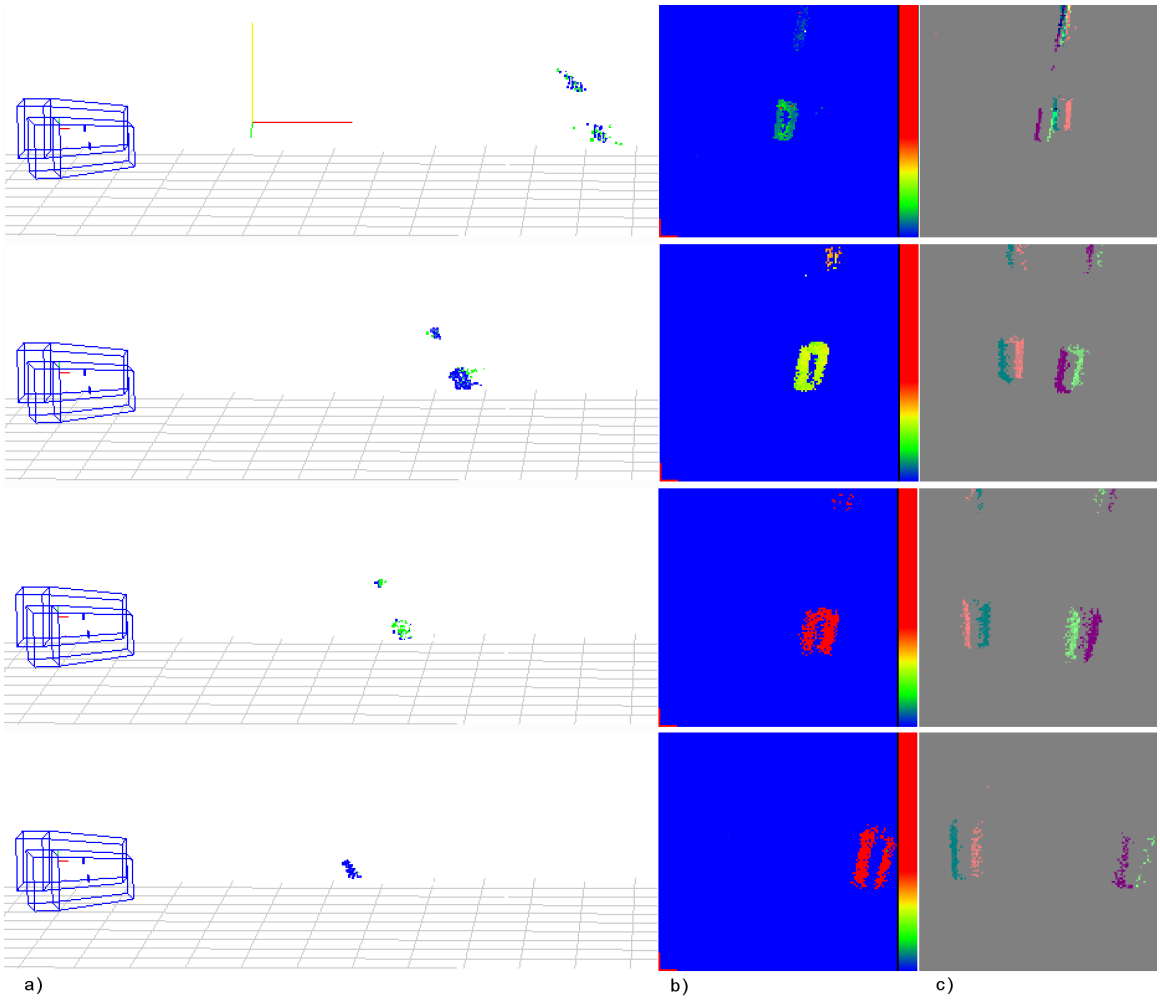

Figure 4.19: a) 3D reconstruction viewed from the side of the scene, 3D axis is shown in the top display, b) disparity map, color code indicates disparity, blue is small disparity (far objects) while red indicates large disparity (closer objects) c) raw data of stereo recording, color coded with green/violet for left ON and OFF camera events, and salmon/cyan right ON and OFF camera events.

3D trajectory of a rotating ball

We repeated the experiment but with a different movement: here the ball is rotating anti-clockwise, describing a circle centered on the top position from which the string is attached. The shape of the trajectory is captured by the 3D reconstruction although the ball is moving out of the field of vision on the sides. Figures 4.23 and 4.24 show the evolution of the depth of the ball over time as it rotates. The data is sampled every $5ms$ here but the actual temporal resolution is below 20 μs.

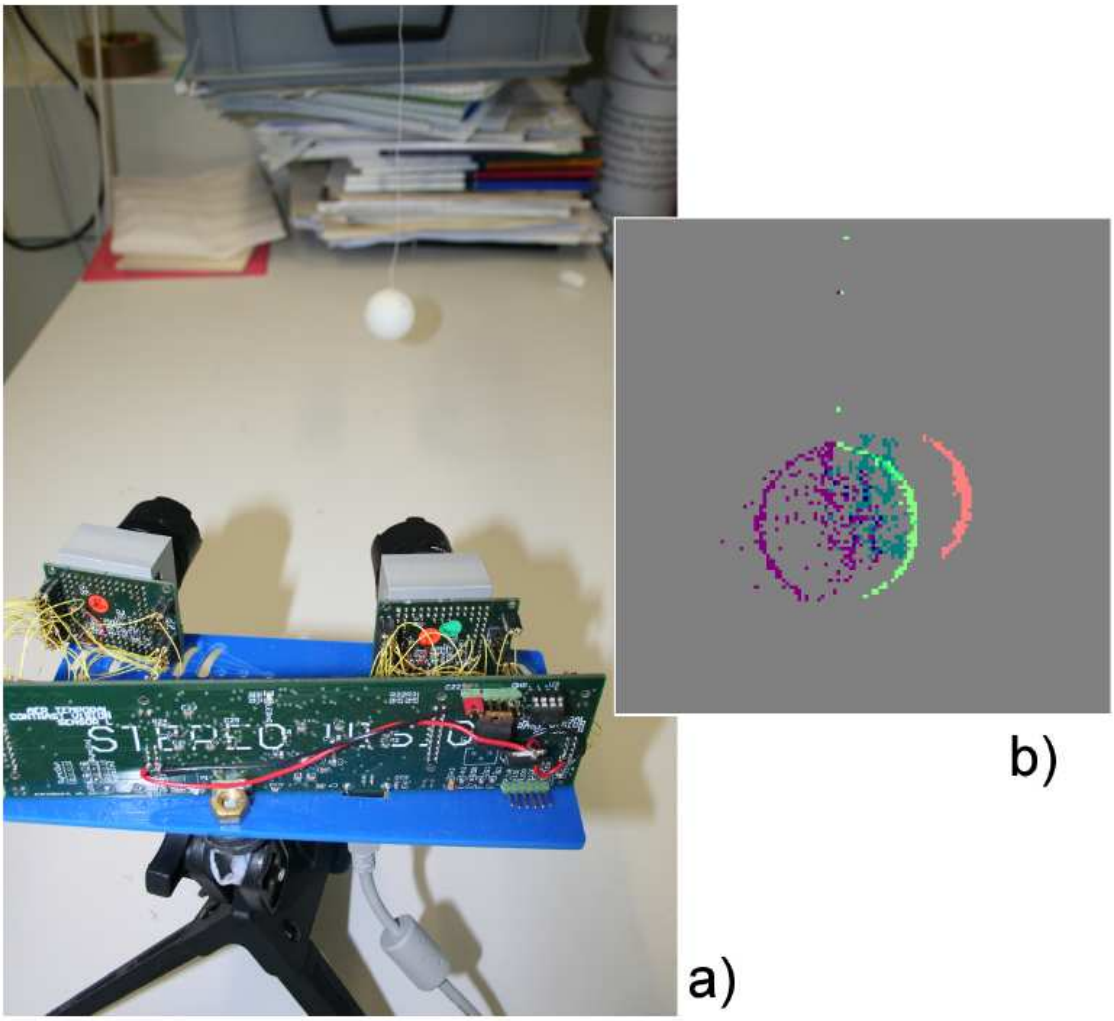

Figure 4.20: a) A white ball is attached to a string and swinged in front of the stereo DVS setup, b) raw DVS data of the moving ball, color coded with green/violet for left ON and OFF events, and salmon/cyan right ON and OFF events..

3D tracking of a two rotating points at different depth

We proceed by testing the coherency of the calibrated stereo matching. The aim is to measure if the distance between two objects is robustly reconstructed independently of the depth of the objects. We fix two points (1cm of diameter each) on a plane and measure the distance between them at 3.30cm. We then present this plane in front of the stereo setup. The plane is mounted perpendicular to a rotor and then rotated at 16Hz. The two points are at 1.5cm and 2cm respectively of the rotation center. This is illustrated in figure 4.25.

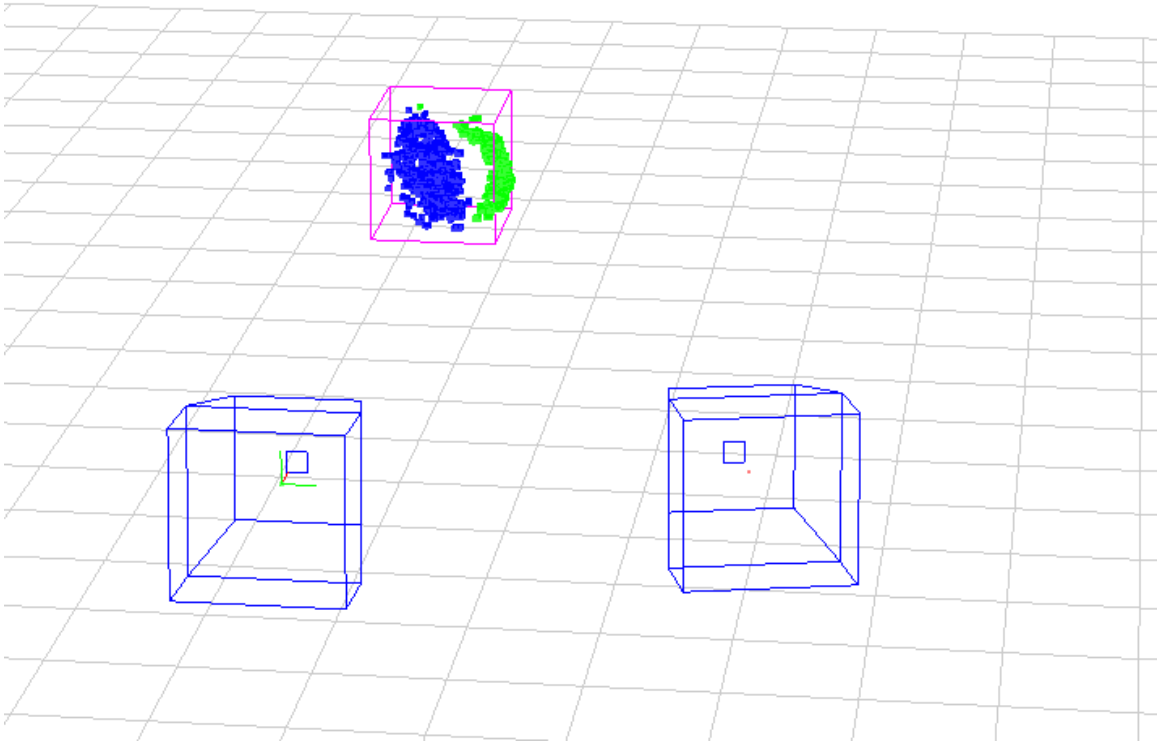

Figure 4.21: The moving ball's 3D events are reconstructed in green (ON events) and blue dots (OFF events), here shown in front of the two camera (in blue lines). The position of the ball is tracked and plotted as a violet cube.

We record the rotating points at three different depths. Figure 4.26 shows the obtained 2D data for all three depths, with a) at around 30cm, b) 40cm and c) 60cm from the left camera. The distance between the two points in the 2D images are respectively a) 44 pixels b) 5 pixels c) -46 pixels, but the distance measured in the 3D reconstruction should be invariably close to the measured 3.3cm. We reconstruct the 3D scene from the stream of 3D events outputted by our event-based stereo matching algorithm, as seen in figure 4.27.
The position of the reconstructed two points is tracked using a mean-shift 3D tracker. The 3D tracker's gravity center is displaced for each 3D events that falls in its volume.
The obtained positions describe neat circles for each obtained depth as shown in figure 4.28 where the position of the outside point is plotted in blue dots while the inner point's positions are shown as red dots.
The distance between the two points is measured every 0.3ms for each depth, over a duration of 3s. Figures 4.29, 4.30 and 4.31 show the measured distance when the points are respectively at 30, 40 and 60 cm from the camera.

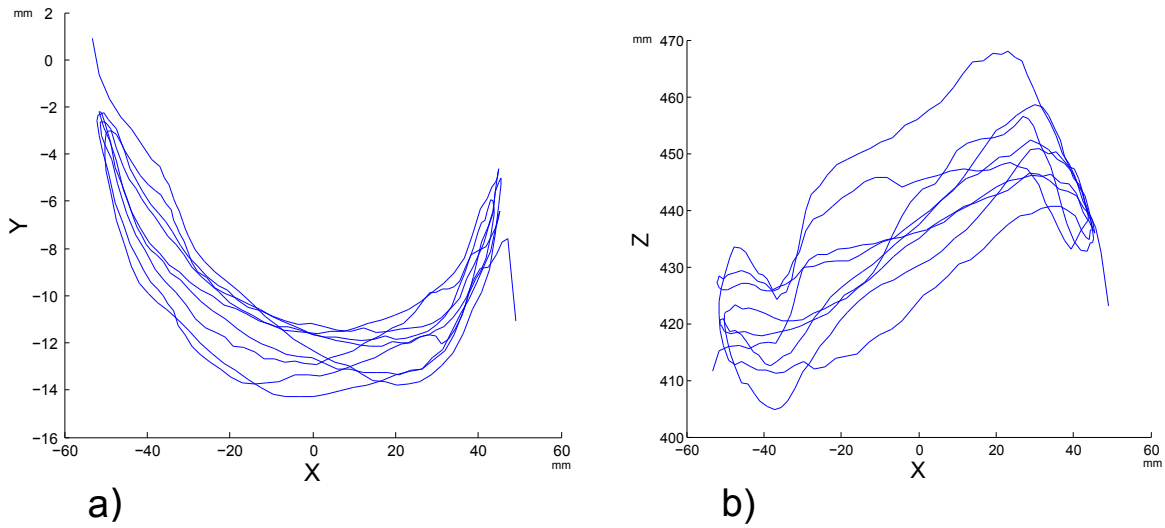

Figure 4.22: a) Swinging trajectory viewed from front, b) trajectories viewed from top. The ball is swung 9 times from left to right and back.

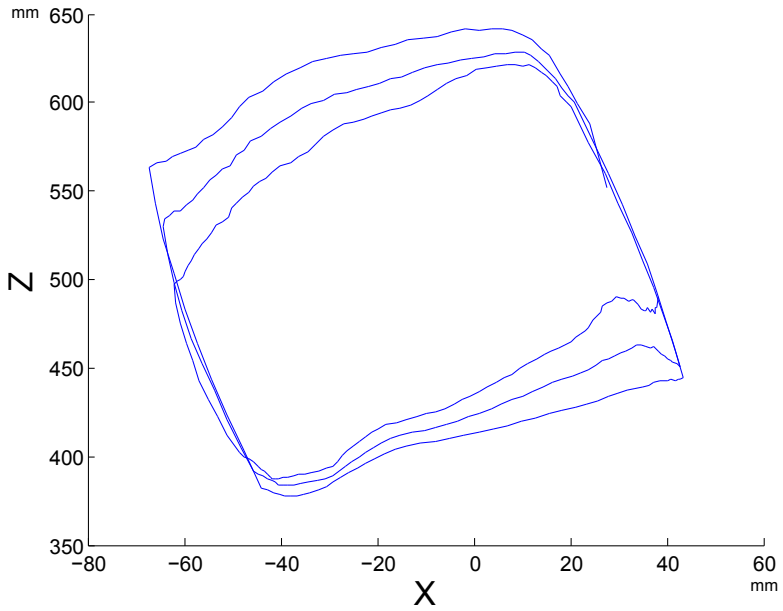

Figure 4.23: Rotating ball trajectory viewed from top. The trajectory is only roughly circular because the ball exits the cameras' field of view on both sides.

The error in the reconstruction of the distance between the two points is plotted for each different depth in figures 4.32,4.33 and 4.34. At 60cm from the cameras, the error is high, this is partly due to the low spatial resolution of the cameras. At 30cm and 40cm the reconstruction error is within 10% of the distance to the

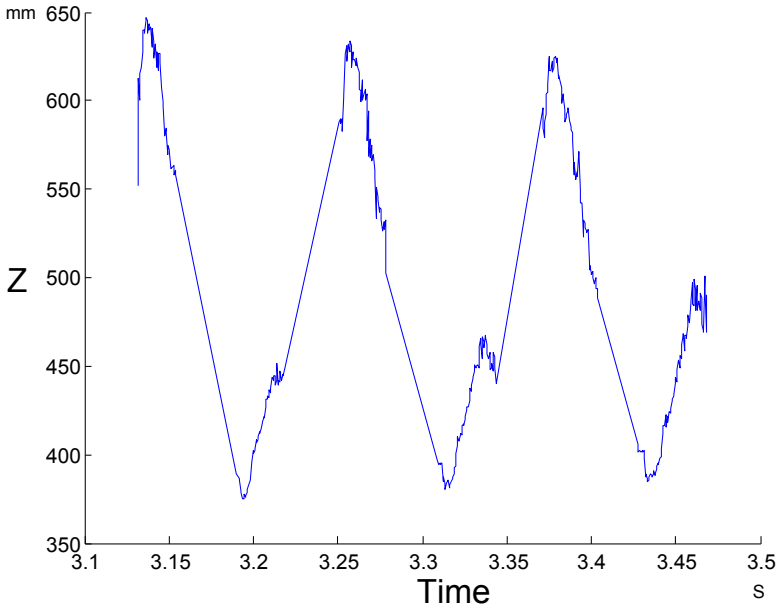

Figure 4.24: Rotating ball depth evolution over time. The temporal resolution is here 5ms but can be selected to be below 20 μs

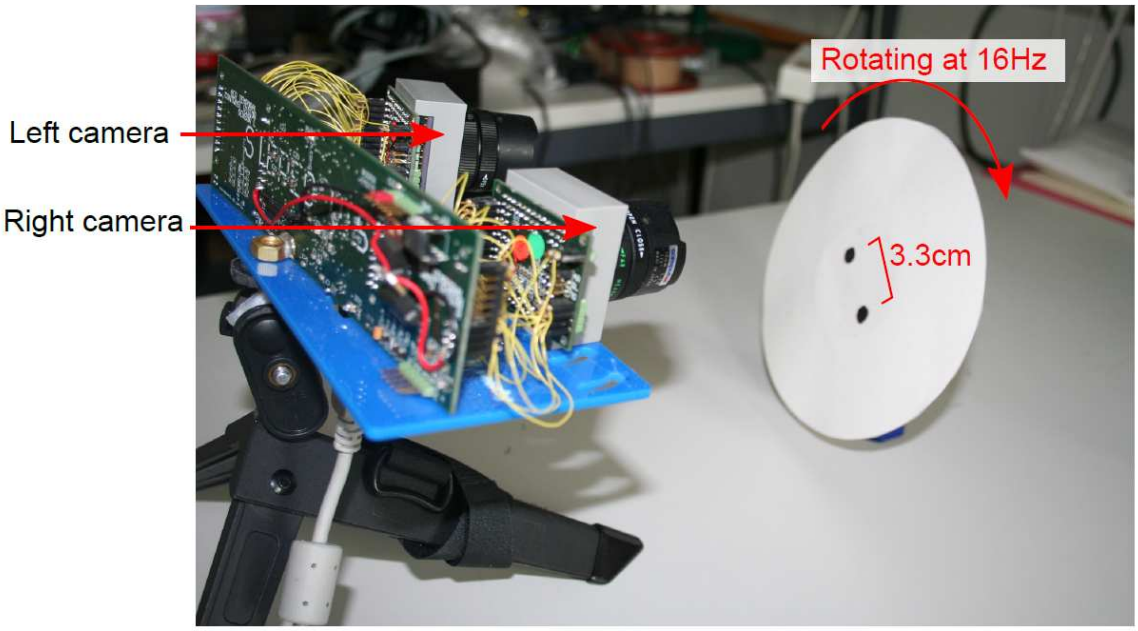

Figure 4.25: Two points are rotating at 16 rotations per second in front of the stereo DVS setup. The distance between the two points is fixed as 3.3cm.

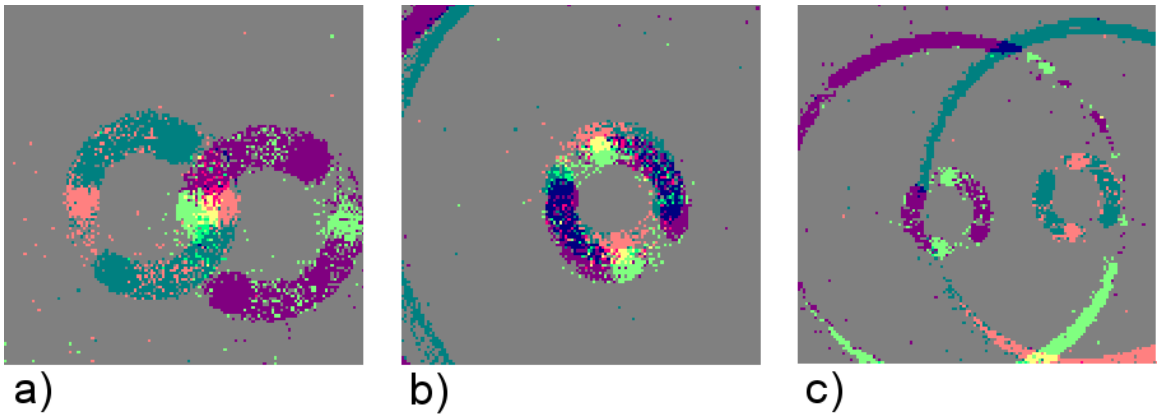

Figure 4.26: DVS raw data, accumulation over 20ms over three different depths, a) 30cm, b) 35cm, c) 40cm. ON and OFF events from both cameras are merged into this display, color coded so that green/violet represent left camera ON and OFF events, and salmon/cyan the right camera ON and OFF events. The distance between the two points in the 2D images is respectively a) 44 pixels b) 5 pixels c) -46 pixels.

camera.
We plot the estimated points distances against the depth of the plane in figure 4.35. We can see that despite the large difference in the 2D data, the reconstructed distance is largely independent of the depth, albeit a slight increase due to the noise in the tracking and reconstruction.

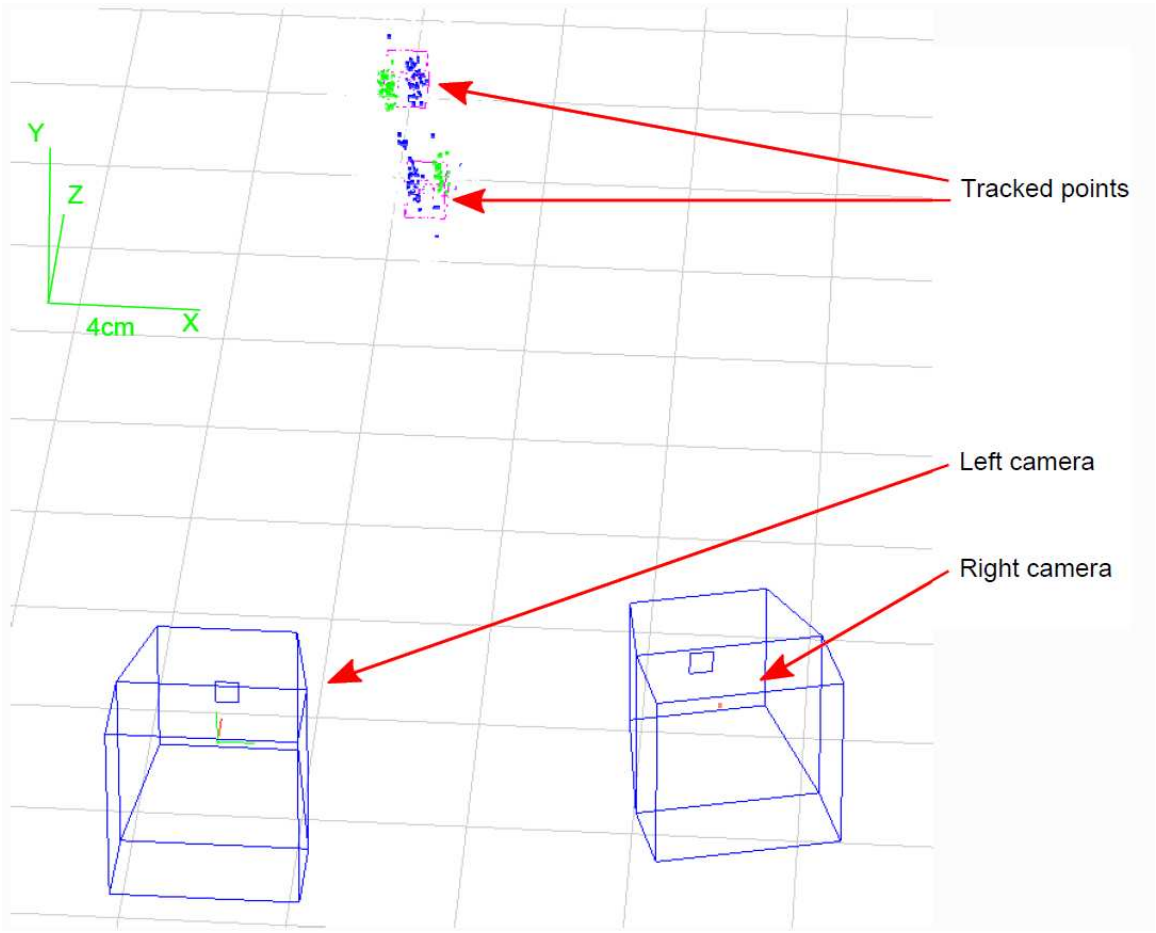

Figure 4.27: 3D reconstruction of the rotating two points in front of the stereo DVS. The positions of the two points are tracked by mean-shift trackers, here shown as violet cubes. Green and blue dots represent respectively ON and OFF 3D events. The cameras are shown in blue and the 3D axis in green lines.

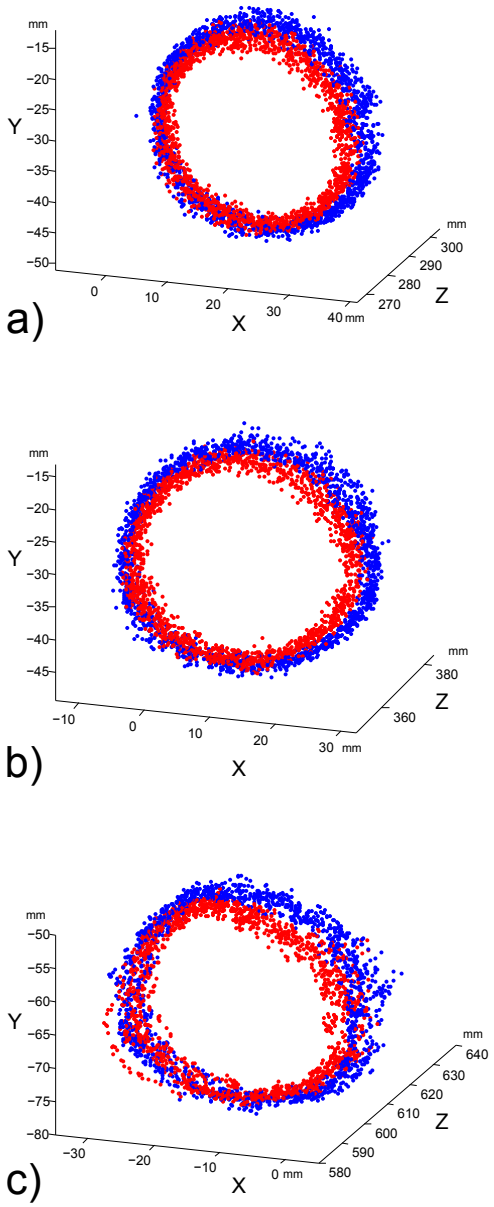

Figure 4.28: 3D positions of each points obtained from the 3D mean-shift tracker, over 550ms. The outside points are marked in blue, the inside points are marked in red. a) at 30cm from left camera, b) 40cm, c) 60cm.

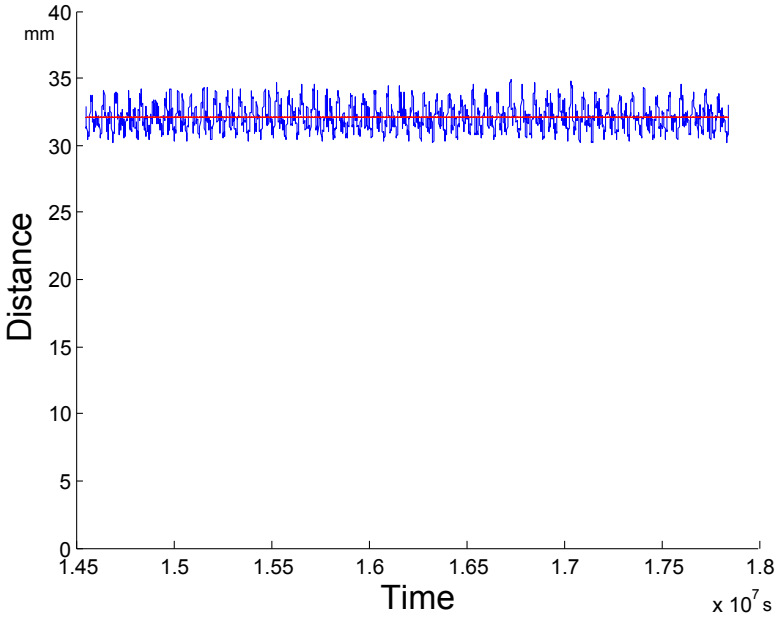

Figure 4.29: Distance, in mm, between two rotating points placed at 30cm from the left camera, measured over time every 0.3ms, during 3s. The mean value is indicated in red.

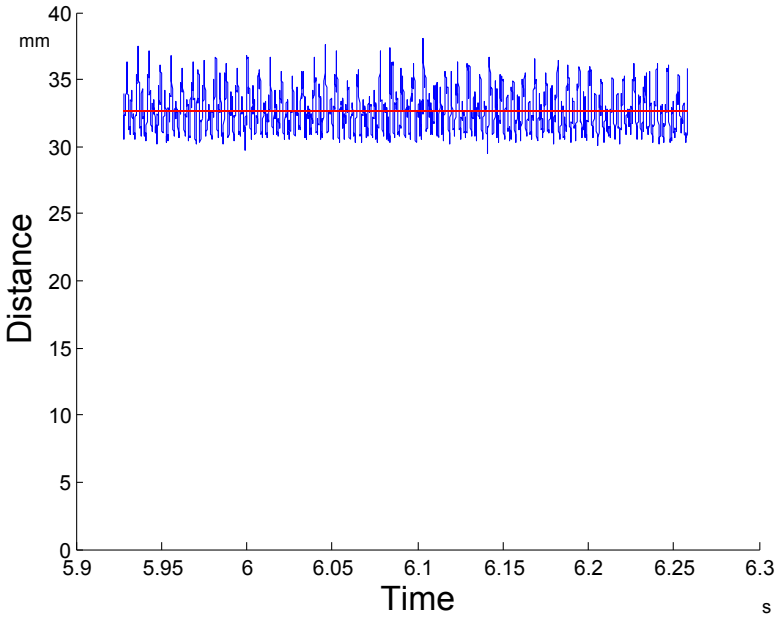

Figure 4.30: Distance, in mm, between two rotating points placed at 40cm from the left camera, measured over time every 0.3ms, during 3s. The mean value is indicated in red.

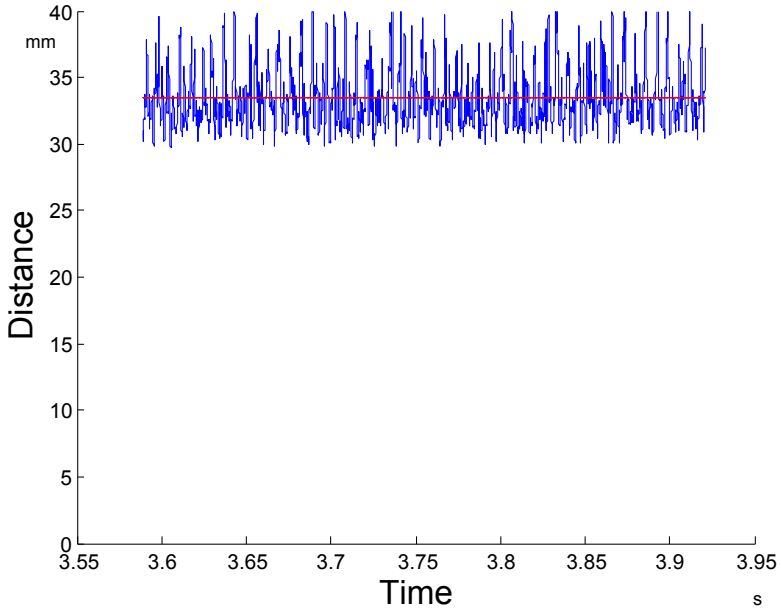

Figure 4.31: Distance, in mm, between two rotating points placed at 60cm from the left camera, measured over time every 0.3ms, during 3s. The mean value is indicated in red.

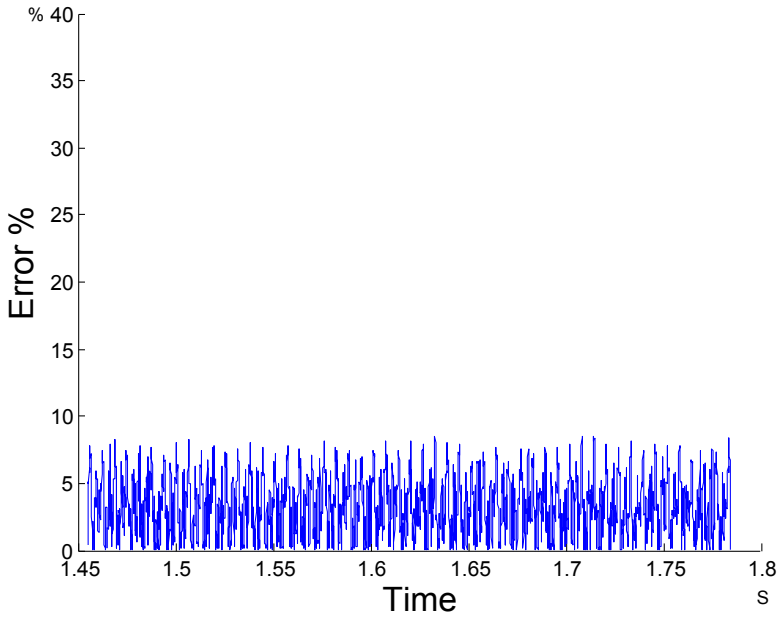

Figure 4.32: Error in the distance estimation, in %, between two rotating points placed at 30cm from the left camera and 3.3cm from each other, measured over time every 0.3ms, during 3s.

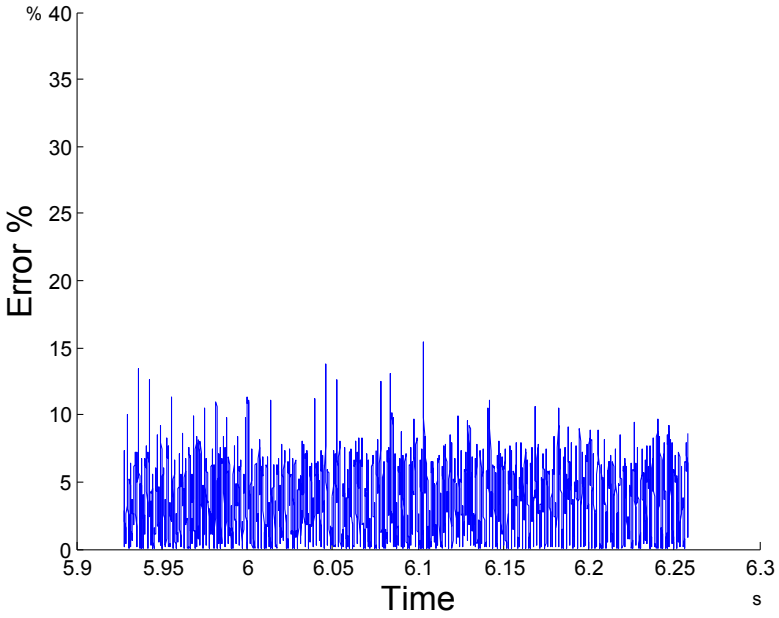

Figure 4.33: Error in the distance estimation, in %, between two rotating points placed at 40cm from the left camera and 3.3cm from each other, measured over time every 0.3ms, during 3s.

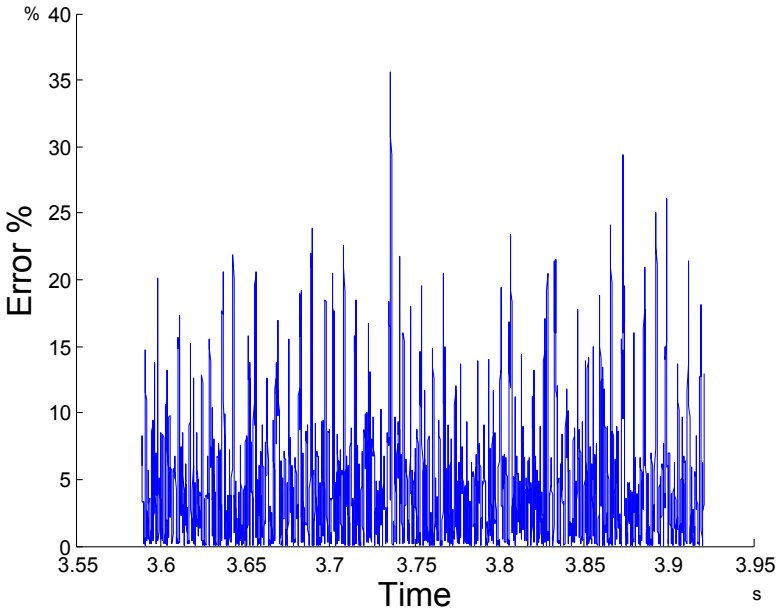

Figure 4.34: Error in the distance estimation, in %, between two rotating points placed at 60cm from the left camera and 3.3cm from each other, measured over time every 0.3ms, during 3s.

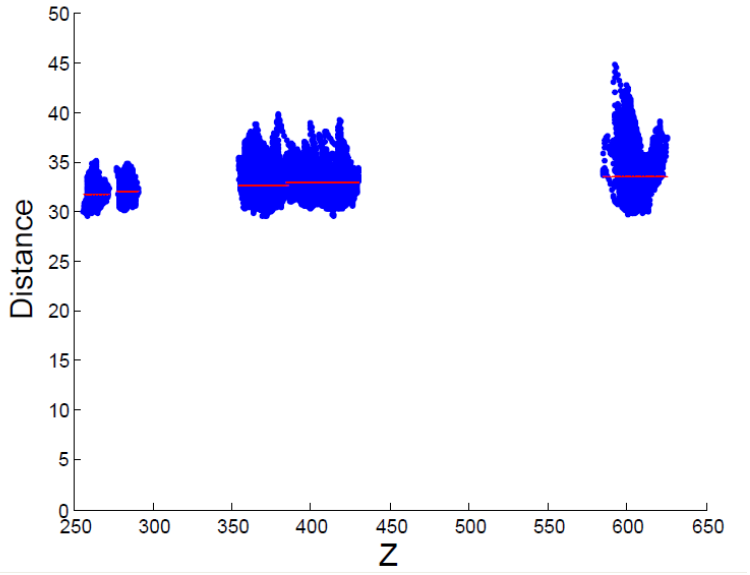

Figure 4.35: Distance, in mm, between rotating two points as a function of the measured reconstructed depth. The obtained distance is close to the measured 3.3cm and is largely independent of the depth of the points.

4.3 Conclusion

The event-based stereo matching algorithm computes on the fly the epipolar line for each left incoming event instead of performing rectification, so as to increase robustness toward calibration errors. The difference between using distances to the epipolar line and rectification is that we can vary the distance threshold to compensate for calibration errors. The algorithm then pre-selects events falling close to a corresponding epipolar line within a given short time window. The correspondence result is obtained by filtering out the few remaining false matches using ordering, polarity, uniqueness and temporal activity constraints. This simple solution proves efficient in matching left and right pixels without the use of extensive spatial computation. This is due to the very high temporal resolution of the camera that allow us to effectively use the distance to the epipolar line as an efficient filter on the potential matches. It is indeed important to note that we do not rectify the camera's view in order to align the epipolar lines orthogonal to the y axis. This rectification is usually employed for simplifying the stereo matching process, as the search for a match can then be limited to scanning the lines $by = 0$ where b is the y coordinates of the left event to match. This rectification especially simplifies the use of spatial convolutions. In frame-based stereo matching, the large number of pixels to match makes it mandatory to reduce the search to a 1D line, but with event-based data, the reduced number of events within the time window acts already as an efficient optimization filter. We can thus increase the tolerance toward the calibration by selecting possible matches within a certain distance of the epipolar line. This approach is made effective by the previous filtering on event's time in a short time window so that only few events are likely candidates. The noise in the result arises from differences in sensitivity between the two cameras setup. It is difficult to set the focus and the camera event-generative thresholds so that as many events are created in both cameras. The choice of the time window's size allows to compromise between matching accuracy and robustness to this sensitivity differences. The time window cannot be chosen too large, as the moving objects will change location and perturb the matching computation. This is due to the nature of the visual event we process here, that only codes for changes in luminance. With a time window ranging from 0.5 to 2ms, the event-based method proves nevertheless efficient with a carefully calibrated and parametrized setup, and is able to perform real-time stereo matching.

Bibliography

[Amini et al., 1990] Amini, A., Weymouth, T., and Jain, R. (1990). Using dynamic programming for solving variational problems in vision. *IEEE Transactions on pattern analysis and machine intelligence*, pages 855–867.

[Arnold and Binford, 1980] Arnold, R. D. and Binford, T. O. (1980). Geometric Constraints in Stereo Vision. *Proceedings, SPIE*, 238.

[Bacon et al., 1998] Bacon, B. a., Villemagne, J., Bergeron, A., Lepore, F., and Guillemot, J. P. (1998). Spatial disparity coding in the superior colliculus of the cat. *Experimental brain research*, 119(3):333–44.

[Baker and Binford, 1981] Baker, H. and Binford, T. O. (1981). Depth from Edge and Intensity Based Stereo. *Proceedings of the 7th international joint conference on Artificial intelligence, Vancouver, BC, Canada*, pages 631–636.

[Banks and Corke, 2001] Banks, J. and Corke, P. (2001). Quantitative Evaluation of Matching Methods and Validity Measures for Stereo Vision. *The International Journal of Robotics Research*, 20(7):512–532.

[Barnard, 1989] Barnard, S. (1989). Stochastic stereo matching over scale. *Internat. J. Comput. Vision*, 3:17–32.

[Barnard and Thompson, 1980] Barnard, S. and Thompson, W. (1980). Disparity analysis of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(2):333–340.

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. *Computer Vision-ECCV 2006*, pages 404–417.

[Bellman, 1957] Bellman, R. (1957). *Dynamic Programming*. Princeton: Princeton University Press.

[Benosman and Devars, 1998] Benosman, R. and Devars, J. (1998). Panoramic stereo vision sensor. In *Proceedings. Fourteenth International Conference on Pattern Recognition*, volume 1, pages 767–769. IEEE Comput. Soc.

[Bensrhair et al., 1996] Bensrhair, A., Mich, P., and Debrie, R. (1996). Fast and automatic stereo vision matching algorithm based on dynamic programming method. *Pattern Recognition Letters*, 17:457–466.

[Brown et al., 2003] Brown, M., Burschka, D., Hager, G., and Others (2003). Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):993–1008.

[Burt and Julesz, 1980] Burt, P. and Julesz, B. (1980). Modifications of the classical notion of Panum's fusional area. *Perception*, 9(6):671–682.

[Canny, 1986] Canny, J. F. (1986). A Computational Approach To Edge Detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8:679–714.

[Cheng and Caelli, 2007] Cheng, L. and Caelli, T. (2007). Bayesian stereo matching. *Computer Vision and Image Understanding*, 106(1):85–96.

[Cooper, 1987] Cooper, P. R. (1987). Order and Structure in Stereo Correspondence. *Technical Report UR CSD / TR 216*.

[Cox et al., 1996] Cox, I., Hingorani, S., Rao, S., and Maggs, B. (1996). A maximum likelihood stereo algorithm. *Computer vision and image understanding*, 63(3):542–567.

[Davies, 2005] Davies, E. (2005). *Basic Image Filtering Operations*, chapter 3. Academic Press, 3 edition.

[Delbruck, 2009] Delbruck, T. (2009). Dynamic Vision Sensor (DVS) - asynchronous temporal contrast silicon retina, http://siliconretina.ini.uzh.ch/wiki/index.php.

[Dhond and Aggarwal, 1989] Dhond, U. and Aggarwal, J. (1989). Structure from stereo-a review. *IEEE Transactions on Systems Man and Cybernetics*, 19(6):1489–1510.

[Gonzalez and Perez, 1998] Gonzalez, F. and Perez, R. (1998). Neural mechanisms underlying stereoscopic vision. *Progress in neurobiology*, 55(3):191–224.

[Gonzalez et al., 1999] Gonzalez, R. C., Cancelas, J. A., Alvarez, J. C., Fernandez, J. A., and Enguita, J. M. (1999). Fast stereo vision algorithm for robotic applications. *IEEE Symposium on Emerging Technologies and Factory Automation, ETFA,*, 1:97–104.

[Hamming, 1950] Hamming, R. W. (1950). Error detecting and error correcting codes. *Bell System Technical Journal*, 26(2):147–160.

[Hannah, 1974] Hannah, M. (1974). *Computer Matching of Areas in Stereo Images*. PhD thesis, Stanford University.

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A Combined Corner and Edge Detection. *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151.

[Hubel, 1959] Hubel, D. (1959). Receptive fields of single neurones in the cats striate cortex. *The Journal of Physiology*, 148:574–591.

[Hubel and Wiesel, 1962] Hubel, D. and Wiesel, T. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106.

[Idris et al., 2009] Idris, M., Arof, H., Tamil, E., Noor, N., and Razak, Z. (2009). Review of feature detection techniques for simultaneous localization and mapping and system on chip approach. *Information Technology Journal*, 8(3):250–262.

[Intille and Bobick, 1994] Intille, S. and Bobick, A. (1994). Disparity-space images and large occlusion stereo. *Computer Vision - ECCV'94*, (220):179–186.

[Jones and Palmer, 1987] Jones, J. and Palmer, L. (1987). An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex. *J. Neurophysiol.*, 58(6):1233–1258.

[Julesz, 1971] Julesz, B. (1971). *Foundations of Cyclopean Perception*. The University of Chicago Press, Chicago.

[Klaus et al., 2006] Klaus, A., Sormann, M., and Karner, K. (2006). Segment-Based Stereo Matching Using Belief Propagation and a Self-Adapting Dissimilarity Measure. *18th International Conference on Pattern Recognition*, pages 15–18.

[Kogler et al., 2009] Kogler, J., Sulzbachner, C., and Kubinger, W. (2009). Bio-inspired Stereo Vision System with Silicon. *7th ICVS (Int. Conference on Computer Vision Systems)*, 5815:174–183.

[Krol and van De Grind, 1982] Krol, J. and van De Grind, W. (1982). Rehabilitation of a classical notion of Panum's fusional. *Perception*, 11:615–619.

[Li and Allinson, 2008] Li, J. and Allinson, N. (2008). A comprehensive review of current local features for computer vision. *Neurocomputing*, 71(10-12):1771–1787.

[Lichtsteiner, 2006] Lichtsteiner, P. (2006). *An AER temporal contrast vision sensor*. PhD thesis, ETH Zurich.

[Lichtsteiner et al., 2008] Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128*128 120dB 15us latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid State Circuits*, 43(2):566–576.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive Image features from scale-invariant keypoints. *International Journal on Computer Vision*, 60(2):91–110.

[Mahowald, 1992] Mahowald, M. (1992). *VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function*. PhD thesis, Pasadena, California.

[Mahowald and Delbrück, 1989] Mahowald, M. and Delbrück, T. (1989). *Cooperative stereo matching using static and dynamic image features*, pages 213–238. Kluwer Academic Publishers, Boston.

[Marr, 1974] Marr, D. (1974). A Note on the Computation of Binocular Disparity in a Symbolic, Low-Level Visual Processor. *MIT AI lab memo 327*.

[Marr, 1982] Marr, D. (1982). *Vision*. Freeman, New York.

[Marr and Poggio, 1979] Marr, D. and Poggio, T. (1979). A computational theoty of human stereo vision. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 204(1156):301–328.

[Ogle, 1953] Ogle, K. (1953). Precision and validity of stereoscopic depth perception from double images. *Journal of the Optical Society of America*, 43:906–913.

[Ohta and Kanade, 1985] Ohta, Y. and Kanade, T. (1985). Stereo by intra-and inter-scanline search using dynamic programming. *IEEE Transactions on pattern analysis and machine intelligence*, 7(2):139–154.

[Ohzawa, 1998] Ohzawa, I. (1998). Mechanisms of stereoscopic vision: the disparity energy model. *Current Opinion in Neurobiology*, 8(4):509–515.

[Parker, 2009] Parker, A. R. (2009). On the origin of optics, Article in Press. *Optics & Laser Technology*.

[Pettigrew and Konishi, 1976] Pettigrew, J. and Konishi, M. (1976). Neurons selective for orientation and binocular disparity in the visual Wulst of the barn owl (Tyto alba). *Science*, 193(4254):675–678.

[Poggio and Fischer, 1977] Poggio, G. F. and Fischer, B. (1977). Binocular interaction and depth sensitivity in striate and prestriate cortex of behaving rhesus monkey. *Journal of Neurophysiology*, 40:1392–1405.

[Qin et al., 2006] Qin, D., Takamatsu, M., and Nakashima, Y. (2006). Disparity limit for binocular fusion in fovea. *Optical Review*, 13(1):34–38.

[Read, 2005] Read, J. (2005). Early computational processing in binocular vision and depth perception. *Progress in biophysics and molecular biology*, 87(1):77–108.

[Rockett, 2003] Rockett, P. I. (2003). Performance assessment of feature detection algorithms: a methodology and case study on corner detectors. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 12(12):1668–76.

[Rodieck and Stone, 1965] Rodieck, R. and Stone, J. (1965). Analysis of receptive fields of cat retinal ganglion cells. *J. Neurophysiol.*, 28:833–849.

[Rosten and Drummond, 2006] Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. *European Conference on Computer Vision*, 1(1):430–443.

[Rosten et al., 2010] Rosten, E., Porter, R., and Drummond, T. (2010). Faster and better: a machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):105–119.

[Scharstein and Blasiak, 2010] Scharstein, D. and Blasiak, A. (2010). Middlebury Stereo Evaluation - Version 2, http://vision.middlebury.edu/stereo/eval/.

[Scharstein and Szeliski, 2002] Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1):7–42.

[Schmid et al., 2000] Schmid, C., Mohr, R., and Bauckhage, C. (2000). Evaluation of interest point detectors. *International Journal of Computer Vision*, 37:151–172.

[Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good features to track. In *1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94.*, pages 593–600.

[Shimonomura et al., 2008] Shimonomura, K., Kushima, T., and Yagi, T. (2008). Binocular robot vision emulating disparity computation in the primary visual cortex. *Neural networks*, 21(2-3):331–340.

[Stewart et al., 1996] Stewart, C., Flatland, R., and Bubna, K. (1996). Geometric constraints and stereo disparity computation. *International Journal of Computer Vision*, 20(3):143–168.

[Tsang and Shi, 2004] Tsang, E. K. C. and Shi, B. E. (2004). A preference for phase-based disparity in a neuromorphic implementation of the binocular energy model. *Neural computation*, 16(8):1579–600.

[Tyler, 1991] Tyler, C. W. (1991). *Cyclopean vision*, pages 38–74. Macmillan Press, Basingstoke.

[Yuille and Poggio, 1984] Yuille, A. and Poggio, T. (1984). A Generalized Ordering Constraint for Stereo Correspondence. *MIT AI Lab AIM-777, Massachusetts Institute of Technology, Cambridge, MA*.

[Zabih and Woodfil, 1994] Zabih, R. and Woodfil, J. (1994). Non-Parametric Local Transforms for Computing Visual Correspondence. *Proc. Third European Conf. Computer Vision*, pages 150–158.

5

Event-Based Monitoring of the Pellet Reaching Task

The loss of the forelimb function is the most disabling handicap for humans. The use of the hands cannot be replaced yet, even partially, by artificial means, and thus considerably harm the autonomy of the victim of such loss. The study of rehabilitation of the forelimb function is an active field where accurate quantification of the forelimb movement is needed to improve on the rehabilitation techniques and studies. But due to the complexity of the evaluation of these movements (high degrees of freedom, specific requirements on hindrance), the current methods for forelimb function monitoring lack the precision and the efficacy that is required. A clear example of this problem is the lack of accurate monitoring methods for the extensively used pellet reaching task in the rat model for forelimb function studies [Maier and Schwab, 2006]. The very fast movement of the rat's paw, on which no marker nor hindrance such as gloves are easy to use, is currently monitored either by simple success/failure ratio count [Whishaw et al., 1993], or by manual and highly time consuming review of frames captured by high speed cameras [Whishaw et al., 2008] [Alaverdashvili et al., 2008]. In the most advanced case, the accuracy of the results is still low as it relies on both subjective definitions of the features, and on the human evaluation of the movement on rating scales, as the one derived from the EshkolWachman Movement Notation [Metz and Whishaw, 2000]. Thus there is an urgent need for an automated system able to accurately extract the trajectories of forelimb movements, and that could be used even on the very fast movement of tiny paws.

5.1 The Pellet Reaching Task Experiment

5.1.1 Context and Experiment

The rat pellet-reaching task [Whishaw and Pellis, 1990] is an animal model for forelimb rehabilitation, where a rat is trained to use his front paw to reach for

an object, in a manner similar to humans. The effects of lesions or treatments on the performances of the rats in this task are potentially very informative on the mechanisms of both disorder and rehabilitation of the hand function in humans. The pellet-reaching task test the grasping ability of the rats using their forepaw. The setup consists of a cage with a narrow opening through which the rat can reach a small pellet of food when extending its forelimb through the opening, see figure 5.1. The size and the height of the opening controls the difficulty of the task for which the animal must be trained. The training lasts typically two weeks with a gradual increase in distance, from inside the cage to the full forelimb extension distance.

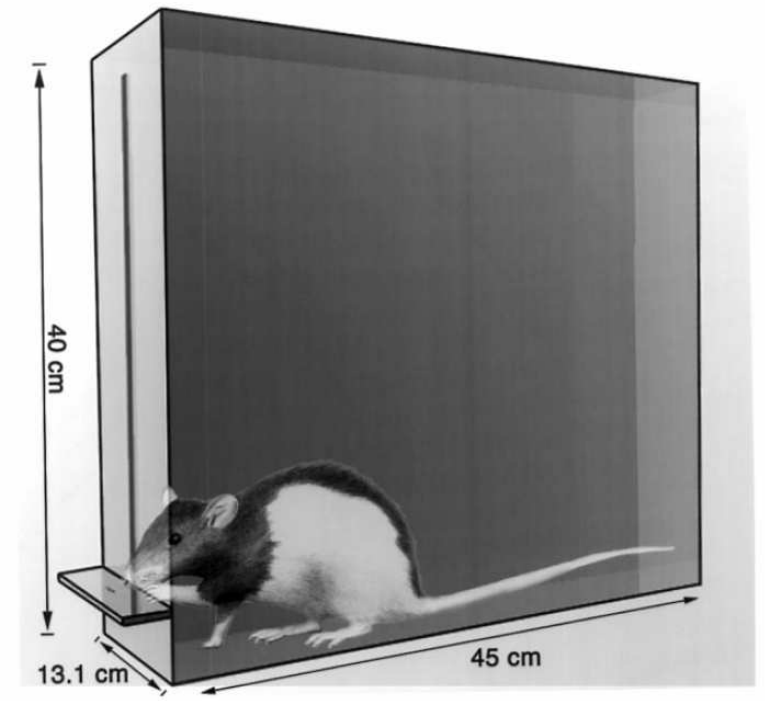

Figure 5.1: The setup consists of a cage with a narrow opening through which the rat can reach a small pellet of food [Metz and Whishaw, 2000].

The paw of the rat is small (15mm long by 10mm wide in average) and the movement is fast, an entire grasp lasting approximately 190ms. The quantification of the performances is thus mostly based on successes count [Bracha et al., 1990] [Whishaw et al., 1993], as the characterization of the movement itself requires high-speed cameras and frame-by-frame analysis by a human observer [Metz and Whishaw, 2000], which is time costly and not accurate enough to characterize the trajectory of the movement [Muir and Webb, 2001], see figure 5.2.
In comparison to the human hand, the movement of the rat's paw is very fast,

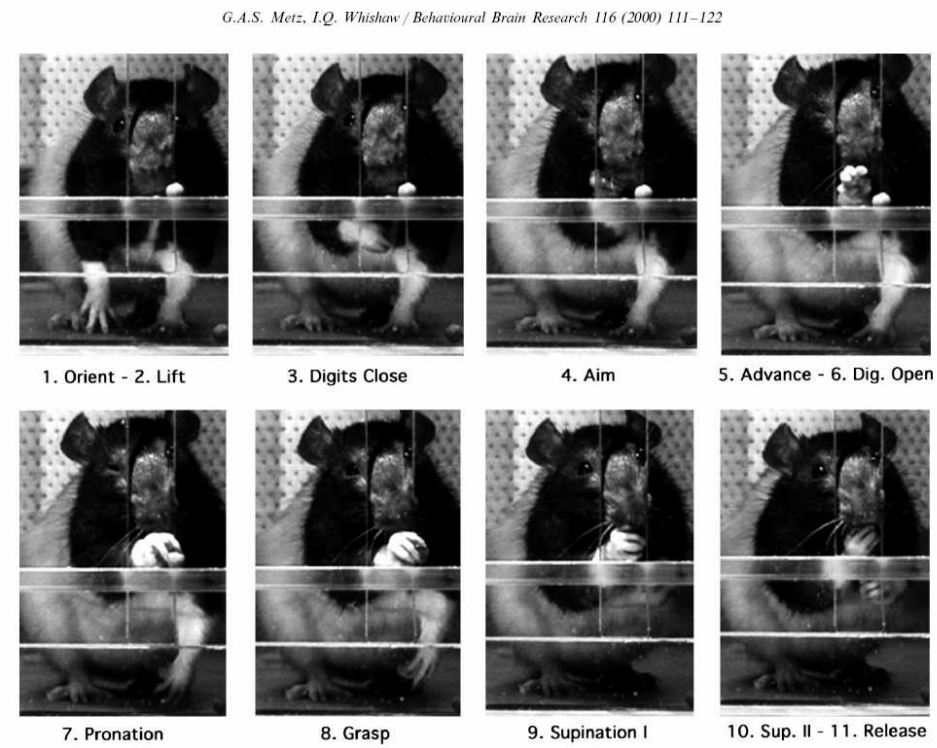

Figure 5.2: Characterization of the grasp movement in the rat pellet-reaching task using frame-by-frame analysis by a human observer [Metz and Whishaw, 2000].

small, and we cannot easily use markers. The existing human hand tracking techniques lack the needed accuracy and robustness in this specific context. There is an urgent need for an automated system able to accurately extract the trajectories of forelimb movements, and that could be used even on the very fast movement of the rat tiny paw.

5.1.2 Tracking Hand Movement

Systems tracking hands face two different problems: extracting the exact trajectory of the hand, and tracking also its conformation, its rotation and the placement of its fingers. Two distinct approaches emerge depending of the placement of markers on the hand or not. The extraction of the trajectory is a general movement tracking problem not limited to the hand. Using one of more cameras leads to the development of 2D or 3D tracking techniques. Sequeira et al. introduced the usage of laser range finders to improve the description of a robot's 3D environment [Sequeira et al., 1995], but these active sensors cannot be used to monitor animal experiments as they are too slow to capture high speed movements.

With a very different design combining laser as a pointer and a photosensor to deduce the position of the reflected laser beam, Cassinelli et al presented a tracking system that can track gestures in real-time [Cassinelli et al., 2005]. But the use of laser is yet too disturbing for the animal model experiments. We thus restrict our tracking problem to the use of vision sensors and describe in the following paragraphs the existing tracking systems using digital cameras as their input. Commercial companies such as PhaseSpace [PhaseSpace, 2010] are selling tracking technologies using active LED markers, with the advantage of identifying each LED independently to improve the resolution of occlusion problems. ARToolkit is a software library, originally developed by Dr. Hirokazu Kato [Kato and Billinghurst, 1999] [ARToolKit, 2010], that supports 3D tracking of black and white patterns. These methods using markers are robust and accurate, and do not always rely on expensive hardware. But tracking the movement of the hand has also been attempted without markers, as, in many applications, imposing the use of markers is a constraint to avoid if possible, even for humans. Shan described a system using a likelihood function to track skin-colored object in real time [Shan et al., 2007]. The reliability of the algorithm depends on the ability to isolate skin-colored objects from background noise, and track them without collision with other skin-colored objects. The results are accurate enough to recognize pre-learned gestures of the human hand, but the noise makes it unreliable to extract exact 2D trajectories. The other aspect of tracking hand movements refers to ability to detect and characterize hand conformation, which includes the position and angles of the fingers. Tracking the hand conformation requires extracting relevant and reliable features and managing multiple hypothesis to deal with self-occlusion and ambiguity. The following approaches are performed without markers: Van Gool, Bray et al. have implemented a complex model of the hand, made of polygons so as to accurately match a real hand shape [Bray et al., 2007]. This 3D model is articulated with 15 degrees of freedom and generates views to be matched with extracted 3D shapes from the input. The exploration of this search space is optimized using a convergence-improved gradient descent algorithm, the stochastic meta descent algorithm [Bray et al., 2005], and smart particle filters, representing the multiple hypothesis. The process is too slow for on-line tracking, but the performances in controlled environment are accurate. But using such a complex model limit the reliability to the extent of the similarity between the real hand and the model, as well as the accuracy of the 3D shape extraction from the input. Stefanov et al. proposed another markerless method, but using a much simpler model [Stefanov et al., 2007]. After background removal and filtering on the skin color, they attempt to extract predefined features, such as knuckles and wrist position, so as to map to it a simple hand model consisting of bones and joints. They use an annealing particle filter to match model points with the extracted features even when some are missing

or obstructed, and a variable length Markov model to weight the mapping based on previously found positions and predefined behavioral constraints. The results are obtained in real-time and very accurate, the limitation being that only a subset of predefined gestures can be tracked. Furthermore these promising methods are effective as long as the input and the feature detection is reliable. The difficulty and limitations of current state-of-the-art approaches in hand tracking, where the possible self-obstructions and many degrees of freedom makes it a challenging task not yet fully resolved in computer vision [McAllister et al., 2002].

5.1.3 Monitoring the Pellet Reaching Task

Using an automated tracking system that could accurately extract the 3D trajectory of the movement would thus be an invaluable asset to research in this domain. Similar experiments are already monitored using computer vision systems, such as hind limb experiments in rats. The aim is to extract the kinematics of the movements, so that healthy, lesioned and rehabilitated animals behaviors can be compared and analyzed. By adding markers on the legs, Varejao et al. described the 2D trajectories of the hind limbs as viewed from the side [Filipe et al., 2006] and compared 2D and 3D techniques for hind limb tracking in [Couto et al., 2008]. The measurement of continuous kinematics is advantageous and preferable to other forms of quantification such as end-points description, movement notation or discrete kinematics, because the assessment results are more detailed, accurate and with no prior bias in defining the movement components. But the methods for tracking continuous kinematics requires expensive materials and time-consuming analysis unless the task can be automated. Only few simple kinematics measures such as the forelimb angle in the rat swimming task can be accurately monitored, and the exact movement of the front limb in many experiments cannot be tracked accurately yet [Muir and Webb, 2001]. Tracking the forelimb of rats, because of their small size and high speed, is even more constrained than human hand tracking. It is also difficult to use markers. The rats do not tolerate well alterations on their front paw, becoming distracted on licking their paw until the mark disappears.
Using a neuromorphic dynamic vision sensor (DVS) to monitor the movement solves most of these limitations. The very high temporal resolution of the DVS allows it to easily capture the fast movements of the paw. Its transient responses (as events) to change of luminance automatically sorts out movement without the need for additional markers or unreliable hue tracking in this experiment where almost only the paw is moving and as long as lightning can be controlled to suppress shadows. With such a sensors the task can be more readily solved if we can efficiently exploit the event-based data of the neuromorphic vision sensors.

5.2 Event-based Monitoring System

We have developed an automated behavioral monitoring system using two neuromorphic Dynamic Vision Sensors (DVS) [Lichtsteiner et al., 2008]. The aim is to take advantage of the DVS characteristics (very high temporal resolution, inherent movement extraction, highly compressed event-based data representation) to monitor the rat's paw during the reaching pellet task. The inherent movement extraction due to the camera's sensitivity to temporal contrast changes refutes the need for costly background extraction and allows focusing the processing on the movement only. The very high temporal resolution of the DVS allows us to capture the very fast movement of the rat's paw, which grasp last 190ms in average. Conventional frame-based cameras should be at least able to record at 200 frames per seconds to follow the grasp, and often limit the recording duration to a few seconds. With the DVS it is possible to record at very high speed for duration exceeding hours, depending on the available hard disk space. The absence of redundant visual information carried by frame-based representation gives us a very compressed data format. A session recording of 30 minutes gives file sizes of around 1 GB, while conventional frame-based high-speed camera would fill the hard drive with 500MB files for 10s recordings. These practical advantages ease the automation of the monitoring process by freeing the experimentalist of the burden of running many short recordings while taking care of the rats. Our setup was furthermore well accepted by the rats. But the main advantage of the DVS recording resides in its event-based output data format, which allow us to use efficient tracking algorithms [Delbruck, 2009].
The ideal desired result would be for each rat, session and condition, to extract the following parameters:

- Velocity, acceleration
- Success rate
- Timing of successes
- Attempt rate
- Number of moves per grasp attempt
- Evolution of trajectories
- Evolution of success rate
- Evolution of rats strategy

5.2.1 Methods

Experiments

We participated in 5 reaching-pellet experiments run by Dr. Irin Maier, Dr. Michelle Starkey and Miriam Gullo at the Brain Research Institute, University of Zurich, during the years 2007 to 2009. The first two experiments were run from 13 February to 12 July 2007 over 7 sessions, and from 11 October to 13 December 2007 to which we participated over three sessions. These experiments were designed to study the effect of various therapies on the rehabilitation of the forelimb function after stroke-like induced impairments. For their main research purposes, the experiments were monitored by a human observer counting the number of successful reaches during a limited period of time, 5 to 10 minutes per rats, with a number of 10 to 20 Long-Evans rats. All experiments were done in two phases: training of healthy rats then surgery to create two groups (lesioned rats and sham-lesioned rats for control).
The next three experiments were run from 11 January to 20 February 2008 over 6 sessions, from 18 April to 4 June 2008 over 6 sessions, and from 17 July 2008 to 8 October 2008 over 9 sessions. Our DVS setup was installed and tried for its design and software development. The rats became used to our presence after a week of task training.

Two different types of cage were used in these experiment: a black panel cage was specifically designed for our purpose, see figure 5.3 and 5.4, and a conventional transparent cage to attempt tracking the movement from the inside of the cage in the last experiment in 2009.

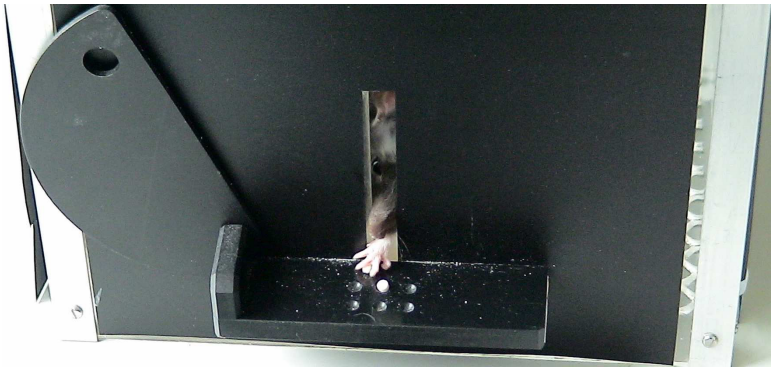

Figure 5.3: A rat is performing a grasp from inside a black panel cage. The contrast of the paw is enhanced but the interior of the cage is not observable.

Figure 5.4: A rat is performing a grasp from inside a black panel cage. Viewed from the side.

Stereo DVS Setup

The setup consists of two DVS mounted in stereo in front of the rat's cage, focusing on the food pellet, see figure 5.5. The setup for the 11/01/08 to 12/02/08 experiment placed the stereo DVS at 15cm from the cage's door with a height difference of 12 cm. The cameras were tilted toward the food pellet at 17 degrees. The lenses used are 12mm lenses put apart by 9.8 cm. The light condition is uncontrolled, only using the ambient light given by a large window. The experiment ran from 9h00am to 11h00am. The rats are put in the cage where they either perform a maximum of 20 grasps or stay 10 minutes maximum. This duration varies across different experiment but not sessions. There were two sessions pre-lesion and 8 sessions post-lesion, the first sessions post-lesion starting a week after lesion. Control and lesioned animals were mixed and their identity hidden from the experimentalist. The experimentalist was taking record of the successful grasp occurrences. Our setup was well accepted by the rats as it was set during their training.
The two DVS have 128 by 128 pixels resolution and their event times are synchronized by an additional synchronization board developed by Patrick Lichsteiner, see figure 5.6. The stereo board output a single stream of combined events that is fed to a laptop running the jAER framework [Delbruck, 2009], see figure 5.7. The laptop we used was a Pentium 4 Laptop connected to the stereo board via USB.

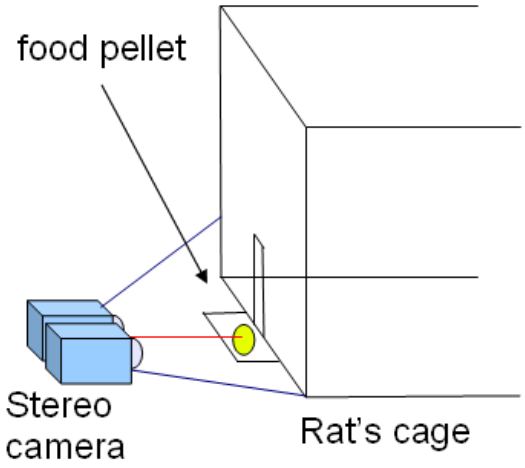

Figure 5.5: We recorded the reaching-pellet task grasps using two DVS of 128 by 128 pixels mounted in stereo in front of the rat's cage.

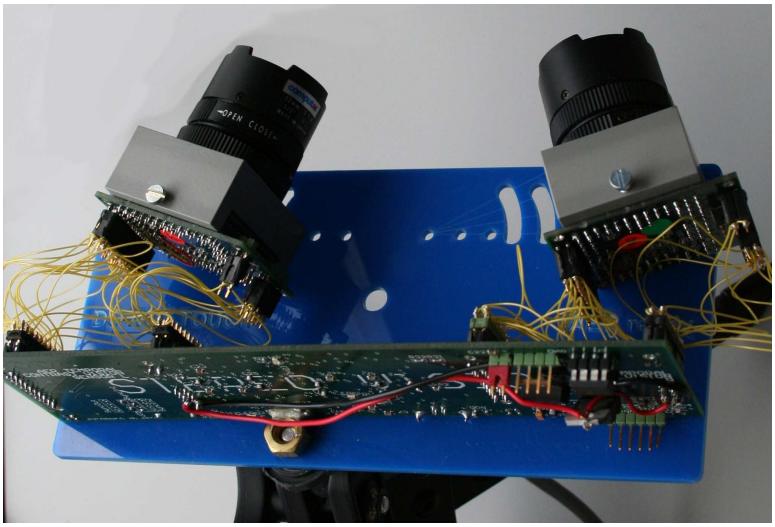

Figure 5.6: The two DVS are mounted in stereo on a laser-cut plastic support and synchronized through an additional synchronization board both designed by Patrick Lichsteiner.

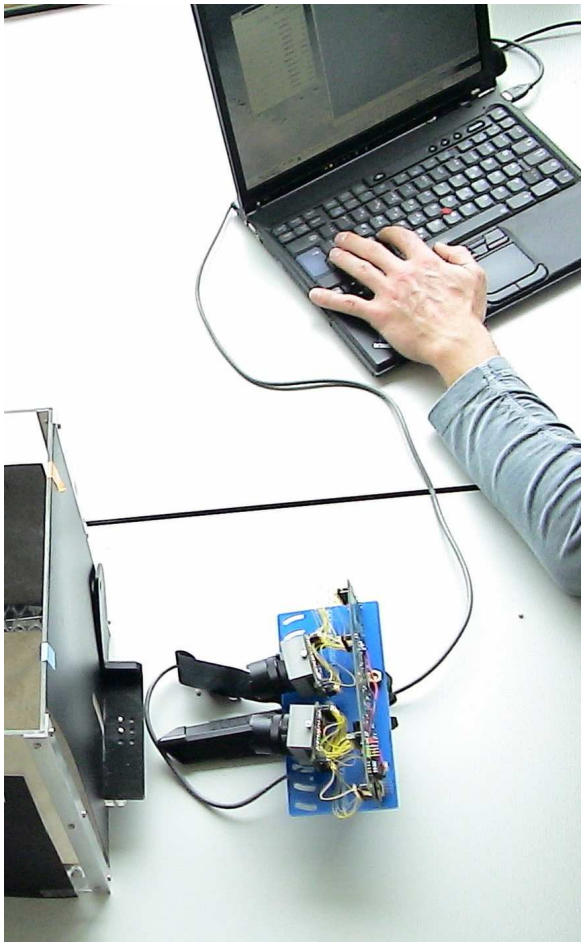

Figure 5.7: The stereo DVS are then placed in front of the cage and connected via USB to a laptop that controls and logs the data.

3D reconstruction

We adapted our stereo matching algorithm developed in chapter 4 to the specific context of this experiment. We perform the matching by enforcing the ordering constraint along with the neighboring constraint. To enhance robustness we take advantage of properties of the controlled setup environment. In particular we use the knowledge about the temporal contrast polarity induced by the moving paw in front of a black panel cage, see figure 5.8.

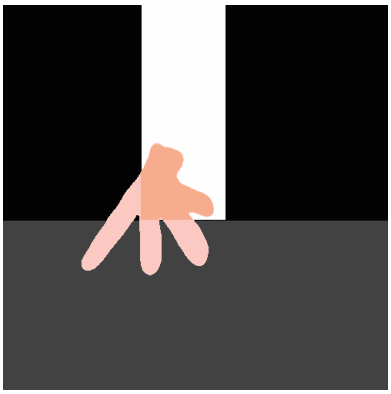

Figure 5.8: The drawing illustrates the polarity of the temporal contrast induced by the rat's paw during a grasp. Using a black panel cage allow to maintain a coherent coding for ON events thus representing the presence of the paw.

We observe that ON events reflects the presence of the paw and OFF events can be either removal of the passing paw or shadows. In the previous chapter, the lifetime of an event was simulated by linear decay so as to create a running time window, see figure 5.9. Here we use the knowledge that a paw event always starts with an ON event when the paw pass in front of the pixel, followed by an OFF event when the paw leaves the pixel's field of view, see figure 5.10.

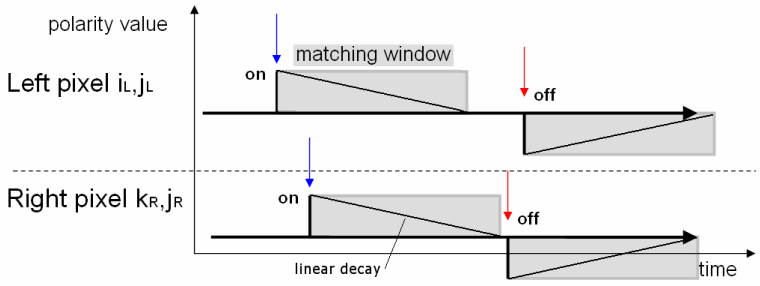

Figure 5.9: Interaction between events is possible through running time windows, here as the event lifetime implemented as linear decay.

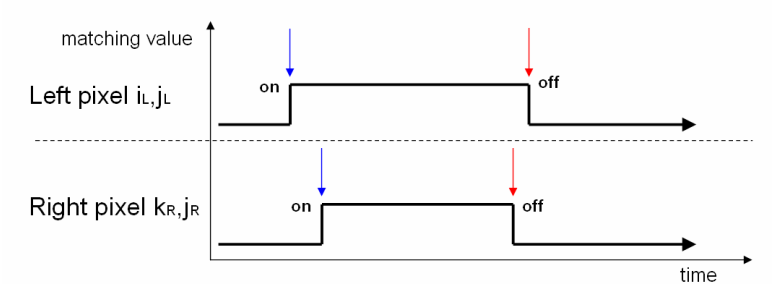

Figure 5.10: The running time windows is bounded by ON and OFF events when contrast coherently represents the presence or absence of an object.

3D Tracking

A tracker, StereoClusterTracker, already exists within jAER that tracks objects from a stereo source by tracking simultaneously the object in the two images. Here we defined a new tracker that operates directly on previously reconstructed 3D events. We adapted the mean-shift tracker developed by Tobi Delbruck [Delbruck, 2009] for event-based data. The mean-shift tracker is defined by equation 5.1, [Fukunaga et al., 1975]. We ported the 2D tracker to 3D by defining a tracker whose gravity center is displaced for each 3D events that falls in its volume, see figure 5.11

$$x \mapsto m(x) = \frac{\sum_{x_i \in N(x)} K(x_i - x) x_i}{\sum_{x_i \in N(x)} K(x_i - x)} \tag{5.1}$$

where $x \mapsto m(x)$ means the tracker position x is mapped to the new position $m(x)$ and $K(x_i - x)$ defines a weight for x_i, typically a Gaussian shaped distance function, but in our case only defined as in equation 5.2, and where $N(x)$ defines neighboring points of x for which $K(x) \neq 0$. We defined $N(x)$ as a cube. The cube's size must be chosen according to the speed and size of the object to track. In our experiment we set it to half the size of the paw for best observed tracking performances.

$$K(x_i - x) = \alpha x + (1 - \alpha) x_i \tag{5.2}$$

In our experiments, α is chosen as 0.1.
The system automatically tracks the paw using the 3D mean-shift tracker and log the data into files. These files are then loaded into the Matlab environment. We have written scripts that automatically smooth and analyses the obtained grasps. The scripts are available in the jAER repository in the folder

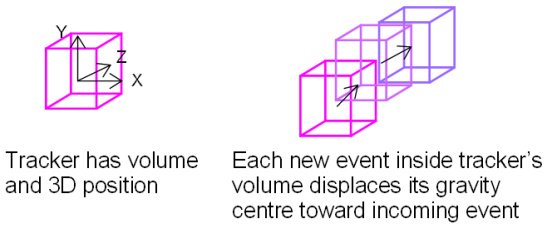

Figure 5.11: A cube volume is used to track incoming 3D events adapting the principles of 2D mean-shift trackers to 3D.

host/matlab/ratmonitoring/pawtracking. The java classes for the tracker is ch.unizh.ini.jaer.projects.stereo3D.Binocular3DTracker.java.

5.2.2 Results

An example of the raw data obtained from our stereo DVS is shown in figure 5.12 where the rat is in the middle of performing a grasp.

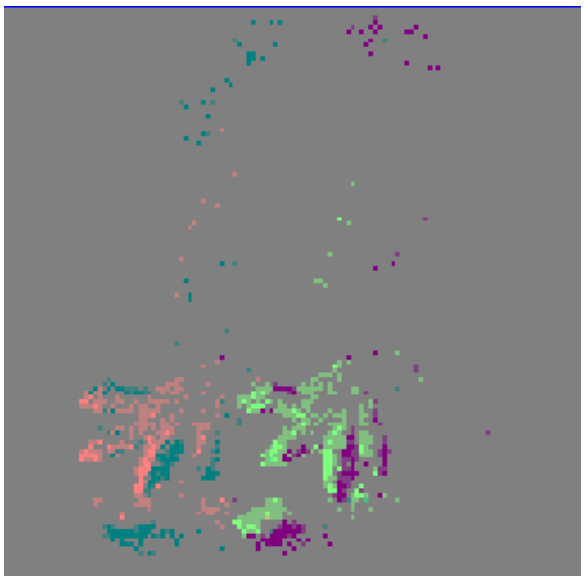

Figure 5.12: Accumulation over 20ms of the stereo DVS response during a grasp. Color codes reflects ON/OFF contrast changes in the two different cameras.

3D reconstruction and tracking

The reconstructed paw and cube tracker are visible in figure 5.13.

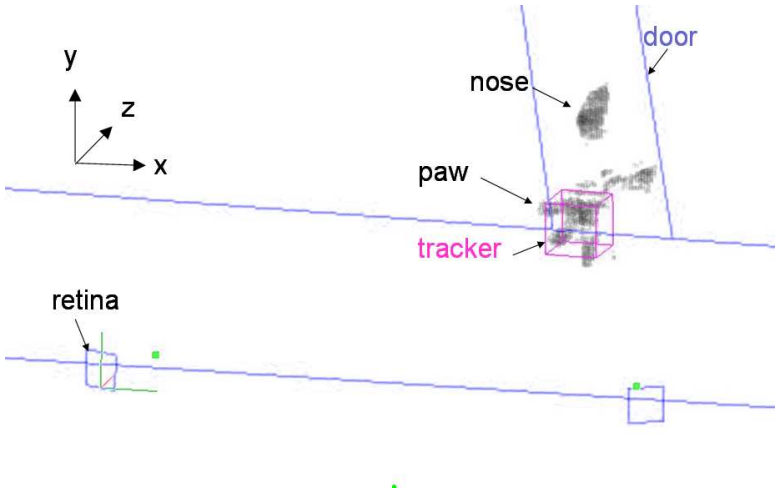

Figure 5.13: Reconstruction and 3D tracking of the paw during grasping. The reconstructed paw points are in grey, the cube tracker following the paw in purple and the cage appears in blue.

The obtained tracker locations are sampled every 1 ms and smoothed to remove noise. The result is visible in figure 5.14 where the recorded position are in blue and the smoothed trajectory in green.

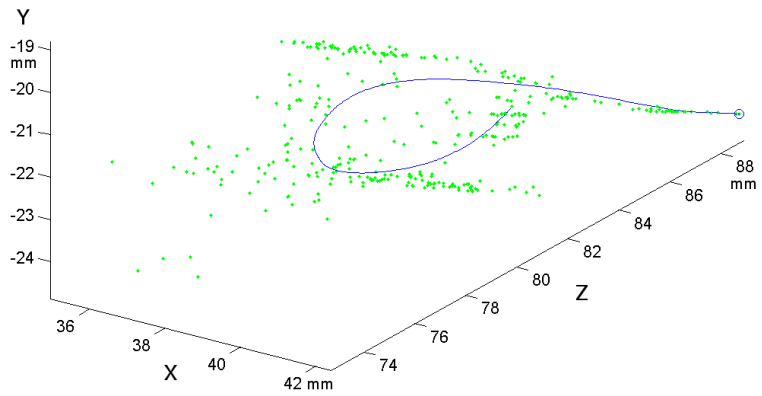

Figure 5.14: Extraction of trajectories by smoothing the recorded tracker position. In blue the tracker's successive locations and in green the smoothed trajectory. Coordinates are in mm.

Trajectories

The label assigned to each rats is below 20 during the training week, but changed to a random number above 20 after lesion. This is done to blind the experimentalist: hiding the identity of lesioned and controlled rats so as to remove bias in the later analysis. Here we plot the smoothed trajectories obtained from rats 4,5,6,10,12 and 15 before lesion (figure 5.15). Both successful (the pellet of food is brought back and eaten) and the unsuccessful grasps are displayed. The orientation of the grasp movement is found to be specific to each rat, despite a different starting position for each grasp.

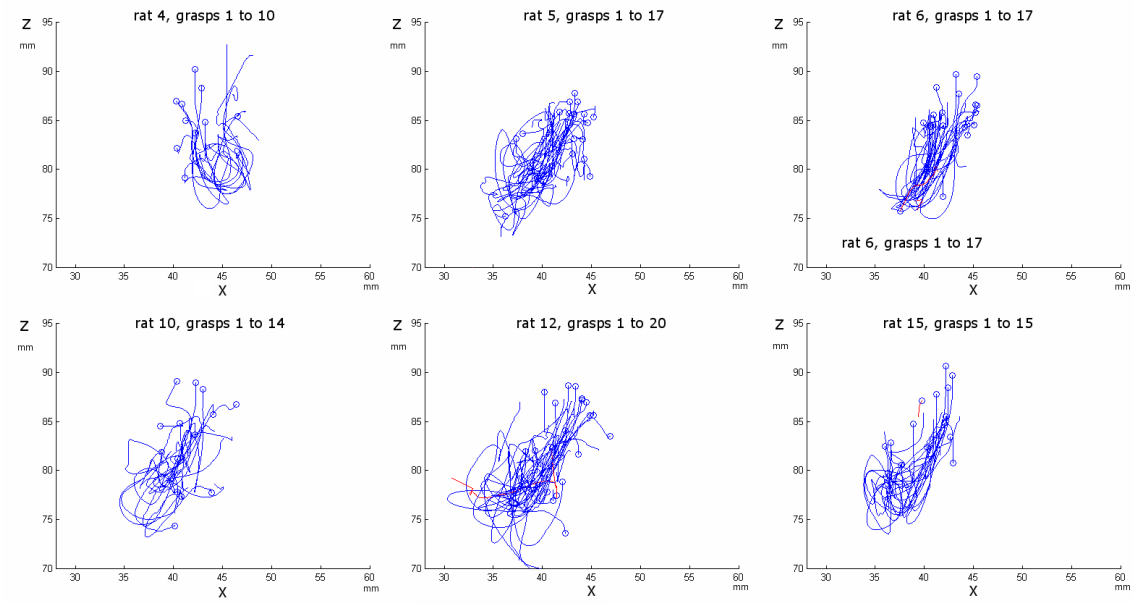

Figure 5.15: View from the top of recorded trajectories for rats 4,5,6,10,12 and 15 after two weeks of training and before lesion, 11 January 2008. Circles indicates the beginning of the tracked trajectories.

We plot here the smoothed trajectories obtained from rats 22 (figure 5.16) 24 (figure 5.17) and 26 (figure 5.18) during the fourth experiment from 11 January to 8 February 2008. These rats are representative of the other 10 rats participating in the experiment. Rat 22 and 26 are left-handed while rat 24 is right-handed. The recovery of the movement is fast, less than a week for most rats, and the movement rehabilitated is found to be similar to the movement before lesion.
To compute the average trajectory per rat and per session we divide the movement in its three x,y and z components. The average height, side and depth displacement are respectively plotted in figures 5.19, 5.20 and 5.21 for the 8 sessions recorded during the experiment running from 17 July 2008 to 8 October 2008.

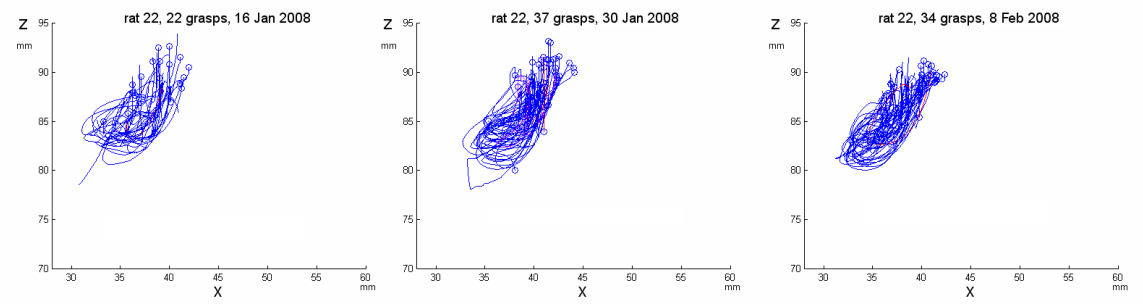

Figure 5.16: View from the top of recorded trajectories for rat 22 after lesioning, over three sessions (16 January, 30 January and 8 February 2008). The coherency in the movement angles and shapes is visible already only four days after surgery.

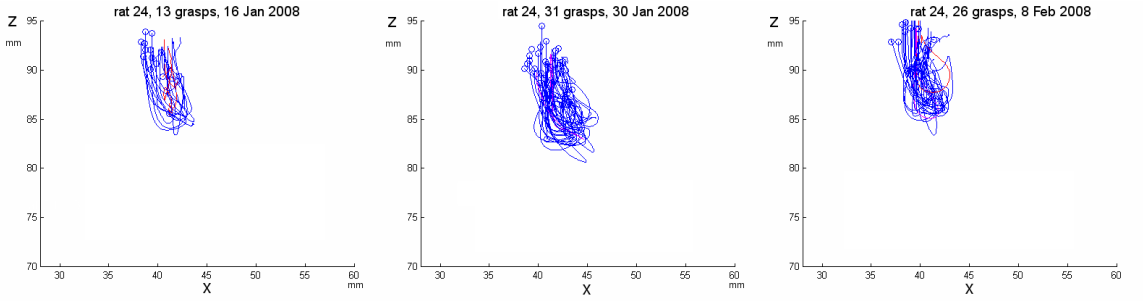

Figure 5.17: View from the top of recorded trajectories for rat 24 after lesioning, over three sessions (16 January, 30 January and 8 February 2008). This rat is right-handed and its movement angle is well captured. Here also the coherency in the movement angles and shapes is already stable less that a week after surgery.

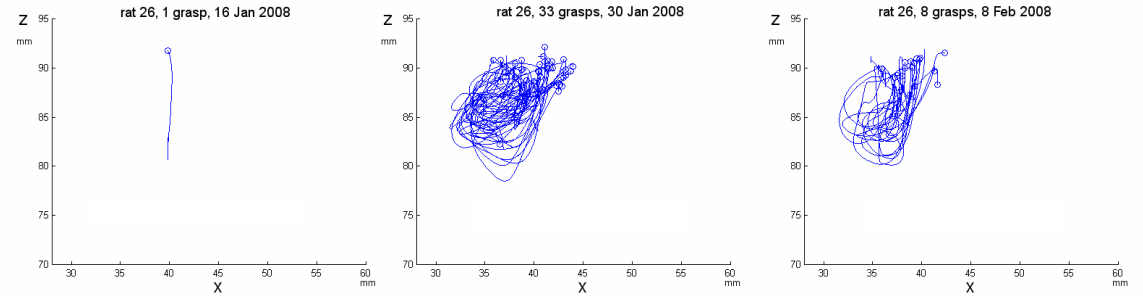

Figure 5.18: View from the top of recorded trajectories for rat 26 after lesioning, over three sessions (16 January, 30 January and 8 February 2008). This rat was unable to perform four days after surgery, but the coherency in the movement angles and shapes came quickly back afterwards.

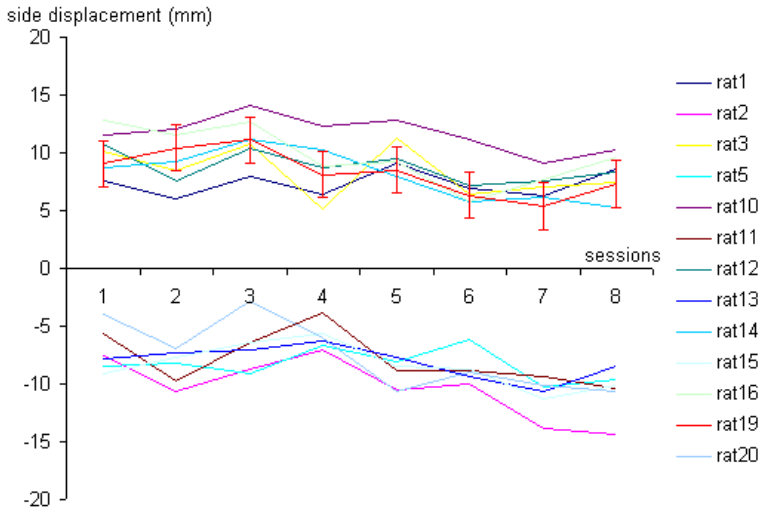

Figure 5.19: Average side extension (X axis component of the grasp movement) for each 13 rats in experiment from 17 July 2008 to 8 October 2008 over 8 sessions. The average standard error (red bars) is representatively given for rat 19. The preference for left and right paw (leading to toward right and toward left grasps) are clearly seen here but otherwise the rats have similar performances.

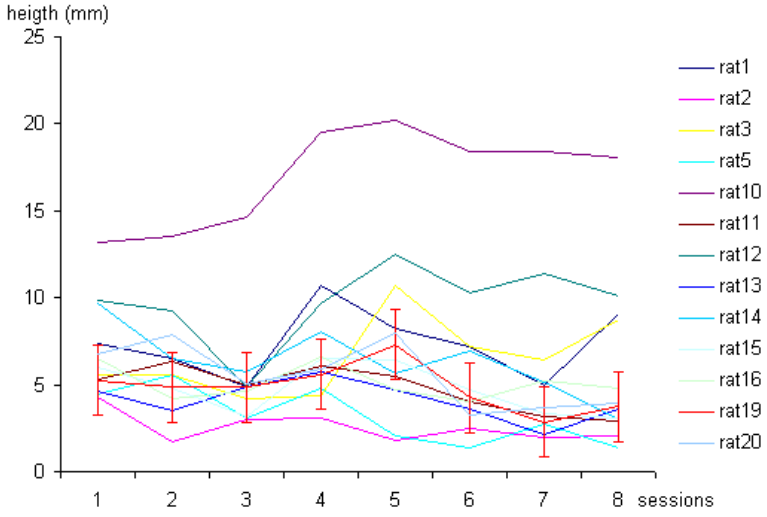

Figure 5.20: Average height extension (Y axis component of the grasp movement) for each 13 rats in experiment from 17 July 2008 to 8 October 2008 over 8 sessions. The average standard error is representatively given for rat 19. One rat, labelled 10, performed unusually high grasps.

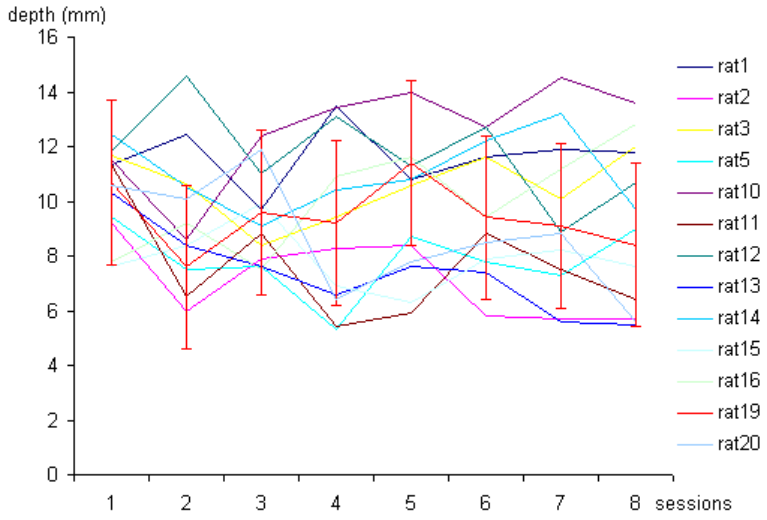

Figure 5.21: Average depth extension (Z axis component of the grasp movement) for each 13 rats in experiment from 17 July 2008 to 8 October 2008 over 8 sessions. The average standard error is representatively given for rat 19. No rat seems to underperform and indeed all were able to grasp as far as the food pellet.

Successful grasps

We did not distinct successful grasps from unsuccessful ones in the previous section. The system is not yet able to automatically detect the difference, but by combining our results with the human-made observation during the task, we can derive statistics about the grasps successes and failures.
The setup is such that a rat must move back toward the back of the cage before the food is replaced and it can try again to grasp the food pellet. But when it is in position, the rat will often try a fast succession of grasps to grab the same food pellet. We call grasp the movement of the paw from the inside of the cage to the food pellet and back in the cage, and we call attempt the ensemble of fast grasps performed for a pellet of food. A rat can produce more than one attempt per pellet if it misses it, but always with a slight pause between attemps, while there is no pause during the fast sequence of grasps that compose an attempt. We recorded the number of attempts and relate it to the number of successful attempts in table 5.1. The evolution over the three sessions of the percentage of successful grasps per attempts is shown in figure 5.22. The amount of successive grasps per attempts is compared with the number of successful attempts in table 5.2.
Although the shape of the grasping movement is quickly recovered, this is not translated into rats' performance in term of successes. The success rate dramat-

	Successes			Total attempts		
rat	16 Jan 08	30 Jan 08	8 Feb 08	16 Jan 08	30 Jan 08	8 Feb 08
22	2	2	4	22	31	18
23	4	9	6	16	20	15
24	1	1	4	8	19	15
25	0	1	3	0	15	24
26	0	0	2	1	21	21
27	0	0	0	1	1	3
28	2	1	4	14	16	20
29	0	0	0	3	15	10
33	2	1	2	14	15	18
34	0	0	0	20	15	18

Table 5.1: Number of successful attempts and total number of attempts.

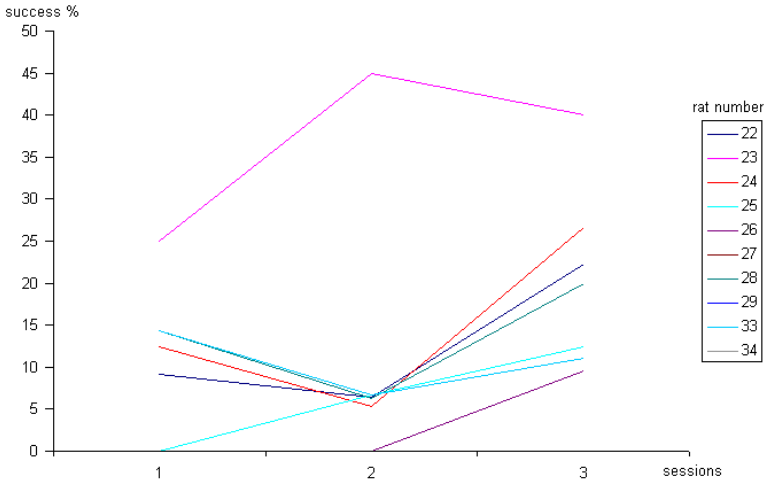

Figure 5.22: Evolution over the three sessions (16,30 January and 8 February 2008) of the percentage of successful grasps per attempts for 10 rats

ically decreases after lesion although the shape of the movement is similar. The movement is less accurate and sometimes much slower. Over the sessions the rats recover and increase their success rates while at the same time reducing the amount of attempts. The ratio of successive grasps per attempts seems stable and depends on the rat. These detailed observation, while requiring the input of an additional human observer for the success count, are more easily observable using the automatic monitoring system and give additional insight on the evolution of the movement recovery.

rat	16 Jan 08	30 Jan 08	8 Feb 08
22	1.50	2.00	2.00
23	1.25	1.44	1.33
24	2.00	2.00	1.25
25	0.00	3.00	2.33
26	0.00	0.00	2.00
27	0.00	0.00	0.00
28	1.00	1.00	1.00
29	0.00	0.00	0.00
33	2.00	2.00	2.00
34	0.00	0.00	0.00

Table 5.2: Ratio of successful grasps per successful attempts.

5.2.3 Limitations

As it stands the system's main limitation resides in the 3D reconstruction accuracy. The current trade-off between robustness and accuracy prevents us to track specific details of the paw such as digits, that would enable us to detect rotations and closure/opening of the paw. Ongoing research on more accurate 3D tracking could solve this problem in the near future. The same problem prevents us to use a transparent cage. The visible body movements add noise to the reconstruction so it is preferable to use black panel cage to increase the robustness of the results. It is thus impossible to track the movement when the paw is inside the cage. Using an additional camera from the top or side to monitor the interior of the cage is a possibility. The detection of successful grasp is not yet implemented. The sensitivity to contrast changes only does not allow to easily detect if the pellet is still in place of was taken away hidden in the rat's paw. The higher spatial resolution of next generation of Dynamic Vision Sensors will be an asset for solving this problem.

5.3 Conclusion

We developed an automatic behavioral monitoring system using stereo Dynamic Vision Sensors and event-based 3D reconstruction and tracking algorithms. Our system has been applied to the monitoring of the reaching-pellet task, from which we could extract the trajectories of the very fast rat's paw. The results show details not observed before, such as the stability of the movement shape in the case of the specific lesion studied in the monitored experiment. The accuracy is yet limited by the spatial resolution of the DVS. The detection of the paw rotation and fingers opening and closure is not yet possible. The use of a black panel cage improves the accuracy of the results but prevent the observations of the movement from inside the cage. The successes and failures of the grasps are still recorded by a human observer, but are easy to integrate with the automatically extracted data. On-going work on the event-based stereo reconstruction shows interesting promises in improving the accuracy, and new neuromorphic vision sensors are developed with higher spatial resolutions that may help solve the current limitations of the system. From the experimentalist's point of view, the system is easy to use, does not disturb the experiment, and allow for long-run recordings at very high temporal resolution. This automatic behavioural monitoring system is the first neuromorphic event-based monitoring system, already showing interesting results. The system stands largely open for improvement, it is important to further its development until all details of the grasps can be automatically recorded.

Bibliography

[Alaverdashvili et al., 2008] Alaverdashvili, M., Leblond, H., Rossignol, S., and Whishaw, I. (2008). Cineradiographic (video X-ray) analysis of skilled reaching in a single pellet reaching task provides insight into relative contribution of body, head, oral, and forelimb movement in rats. *Behav Brain Res.*, 192(2):232–47.

[ARToolKit, 2010] ARToolKit (2010). ARToolKit, http://www.hitl.washington.edu/artoolkit/.

[Bracha et al., 1990] Bracha, V., Zhuravin, I., and Bures, J. (1990). The reaching reaction in the rat: A part of the digging pattern? *Behavioural Brain Research*, 36(1-2):53–64.

[Bray et al., 2005] Bray, M., Koller-Meier, E., Mueller, P., Schraudolph, N. N., and Van Gool, L. (2005). Stochastic optimisation for high-dimensional tracking in dense range maps. *IEE Proc. Vision, Image & Signal Processing*, 152(4):501–512.

[Bray et al., 2007] Bray, M., Kollermeier, E., and Vangool, L. (2007). Smart particle filtering for high-dimensional tracking. *Computer Vision and Image Understanding*, 106(1):116–129.

[Cassinelli et al., 2005] Cassinelli, A., Perrin, S., and Ishikawa, M. (2005). Smart laser-scanner for 3D human-machine interface. In *CHI'05 extended abstracts on Human factors in computing systems*, page 1139. ACM.

[Couto et al., 2008] Couto, P. a., Filipe, V. M., Magalhães, L. G., Pereira, J. E., Costa, L. M., Melo-Pinto, P., Bulas-Cruz, J., Maurício, A. C., Geuna, S., and Varejão, A. S. P. (2008). A comparison of two-dimensional and three-dimensional techniques for the determination of hindlimb kinematics during treadmill locomotion in rats following spinal cord injury. *Journal of neuroscience methods*, 173(2):193–200.

[Delbruck, 2009] Delbruck, T. (2009). Dynamic Vision Sensor (DVS) - asynchronous temporal contrast silicon retina, http://siliconretina.ini.uzh.ch/wiki/index.php.

[Filipe et al., 2006] Filipe, V. M., Pereira, J. E., Costa, L. M., Maurício, A. C., Couto, P. a., Melo-Pinto, P., and Varejão, A. S. P. (2006). Effect of skin movement on the analysis of hindlimb kinematics during treadmill locomotion in rats. *Journal of neuroscience methods*, 153(1):55–61.

[Fukunaga et al., 1975] Fukunaga, K., , and Hostetler, L. D. (1975). The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition. *IEEE Transactions on Information Theory*, 21(1):32–40.

[Kato and Billinghurst, 1999] Kato, H. and Billinghurst, M. (1999). Marker tracking and hmd calibration for a video-based augmented reality conferencing system. *Proceedings of the 2nd International Workshop on Augmented Reality (IWAR 99)*, page 85.

[Lichtsteiner et al., 2008] Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128*128 120dB 15us latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid State Circuits*, 43(2):566–576.

[Maier and Schwab, 2006] Maier, I. and Schwab, M. (2006). Sprouting, regeneration and circuit formation in the injured spinal cord: factors and activity. *Philos Trans R Soc Lond B Biol Sci.*, 361(1473):1611–34.

[McAllister et al., 2002] McAllister, G., McKenna, S., and Ricketts, I. (2002). Hand tracking for behaviour understanding. *Image and Vision Computing*, 20(12):827–840.

[Metz and Whishaw, 2000] Metz, G. A. S. and Whishaw, A. Q. (2000). Skilled reaching an action pattern: stability in rat (Rattus norvegicus) grasping movements as a function of changing food pellet size. *Behavioural Brain Research*, 116(2):111–122.

[Muir and Webb, 2001] Muir, G. D. and Webb, A. A. (2001). Assessment of behavioural recovery following spinal cord injury in rats. *European Journal of Neuroscience*, 12(9):3079–3086.

[PhaseSpace, 2010] PhaseSpace (2010). PhaseSpace, Inc, http://www.phasespace.com/.

[Sequeira et al., 1995] Sequeira, V., Goncalves, J. G. M., and Isabel Ribeiro, M. (1995). Autonomous Systems 3D environment modelling using laser range sensing. *Robotics and Autonomous Systems*, 16:81–91.

[Shan et al., 2007] Shan, C., Tan, T., and Wei, Y. (2007). Real-time hand tracking using a mean shift embedded particle filter. *Pattern Recognition*, 40(7):1958–1970.

[Stefanov et al., 2007] Stefanov, N., Galata, A., and Hubbold, R. (2007). A real-time hand tracker using variable-length Markov models of behaviour. *Computer Vision and Image Understanding*, 108(1-2):98–115.

[Whishaw et al., 2008] Whishaw, I., Whishaw, P., and Gorny, B. (2008). The structure of skilled forelimb reaching in the rat: a movement rating scale. *J Vis Exp.*, 18:816.

[Whishaw and Pellis, 1990] Whishaw, I. Q. and Pellis, S. M. (1990). The structure of skilled forelimb reaching in the rat: A proximally driven movement with a single distal rotatory component. *Behavioural Brain Research*, 41(1):49–59.

[Whishaw et al., 1993] Whishaw, I. Q., Pellis, S. M., Gorny, B., Kolb, B., and Tetzlaff, W. (1993). Proximal and distal impairments in rat forelimb use in reaching follow unilateral pyramidal tract lesions. *Behavioural brain research*, 56(1):59–76.

6

Conclusions

6.1 Event-based Algorithms

In chapter 2 we introduced the notion of event-based vision algorithm. The event-based algorithms are designed to process spike-like event data representation. In the specific context of vision, an event is the asynchronous detection of a change in the visual scene. Emulating the transient response of the biological retina, an event is a relative change of intensity as perceived by the asynchronous neuromorphic pixel. Thus an image is not encoded as a frame of synchronous intensity readings, but as a stream of events. This data supports different computation schemes than frames by making full use of the temporal dimension. The first challenge was to perform computer vision on this new data representation, while preserving the experience accumulated by the computer vision community.

We transformed two frame-based filters into event-based vision filters, the spatial low pass filter and the thinning filter. We defined a spatio-temporal convolution that operates locally in time for each incoming event, achieving similar results as the frame-based filter. The thinning filters also performs a spatial operation, reducing thick segments to their thinnest structure. We showed how to adapt a parallel frame-based version of the thinning algorithm. The conversion is straightforward as parallel algorithms already perform local computations. We showed that classical computer vision approaches can be adapted to event-based processing. The event-based filters were implemented and run on the spike-like input of a Dynamic Vision Sensor and compared favorably to frame-based filters.

6.2 Event-based Stereo Vision: Calibration

We moved toward a more complex vision task by addressing stereo vision. The mandatory calibration of the sensors is explained and adapted to event-based

sensors in chapter 3. The calibration is the process that estimates the cameras intrinsic and extrinsic parameters. The possibility to use frame-based established methods was demonstrated. Epipolar rectification was adapted to event-based data where the extrapolation of neighboring values takes the form of a probability distribution of incoming events. We performed the calibration and rectification of a stereo rig of Dynamic Vision Sensors so that we can address the problem of stereo correspondence in the following chapter.

6.3 Event-based Stereo Vision: Correspondence Solving

Stereo vision is the process of estimating the depth of objects in the visual 3D scene from two cameras. The projection of the object's 3D points onto the two cameras is dependent of the object's distance. The difference in the position of pixels representing the 3D points in the two views is the disparity. Knowing the disparity in its pixel's projection, it is possible to recompute any 3D point's location by triangulation. The main difficulty is to find which pixels are representing the same 3D point in the two views. This problem is called the stereo vision correspondence problem. In chapter 4 we present an event-based approach that differs from previous computer vision solutions. Our event-based stereo matching algorithm uses the temporal dynamics of events to simplify the matching process. It depends on the setup calibration but is more robust to calibration errors than methods using epipolar rectified images. By exploiting the timing of incoming events and a few constraints it is able to estimate the disparity of moving objects.

6.4 Event-based Monitoring of the Pellet Reaching Task

In chapter 5, we applied our solution in event-based stereo vision to the monitoring of behavioural experiments. We designed an automatic system that quantifies the movement of the rat's paw in the rat pellet-reaching experiment. The pellet-reaching experiment is used to study the rehabilitation of hand function. Rats are trained to perform forelimb movements that are then characterized to assess the efficacy of rehabilitation techniques. But this characterization currently lacks accurate quantification of the movement, due to the difficulties in tracking the fast and marker-less paw. We succeeded in implementing a system using two neuromorphic dynamic vision sensors (DVS) and event-based algorithms to track the

rat's paw. Our system is easy to use and set up, records compressed data for long runs and at high temporal resolution. The results are accurate enough to extract 3D trajectories of the paw and quantify the evolution of the movement over sessions. The additional input from human observers complements the information given by the system that records all attempts at grasping, successful or not. Important details like the rotation, opening and closing of the paw, as well as the beginning and end of the movement inside the cage are yet difficult to obtain, but the potential for improvement of our prototype shows that the system is promising in such applications.

6.5 Conclusion

The key question of this thesis was how to use the particular spike-like data representation provided by neuromorphic sensors in the context of vision. We showed in chapter 2 that we could reuse the experience accumulated by the computer vision community despite its implicit focus on frame-based representation. In chapter 3 and 4 we addressed the complex problem of stereo vision both by adapting frame-based techniques and by developing new event-based methods taking advantage of the event data representation and its temporal dynamics. These event-based solution can be already implemented to start solving difficult tasks such as behavioural monitoring as we have shown in chapter 5.

6.6 Discussion

6.6.1 Event-based computer vision

The classical representation for an image is frame-based, i.e an array of intensity values measured at the same time. Most cameras use this representation and the long developed experience of the computer vision field is largely based on the processing of frames. But recent progress in neuromorphic engineering allow us to process visual data in the event-based representation that is inspired from the biological retina. The change of representation from frames to events is not only a matter of storage or compression of the data, it provides also new computational possibilities.
Exploiting these new computational possibilities is not a trivial task. The event data is particularly adapted to the processing of motion, but the question remained about its ability to encode and process spatial vision. So although efficient algorithms have been developed to process motion related tasks such as

tracking, the first reflex when attempting to perform spatial computer vision on event-based data was to convert it first to frame-based data. One of the result of this thesis is thus to propose new principles for building event-based vision algorithms. Event-based vision is distinct from classical computer vision in these two respects: the data is different as it encodes temporal information that can be exploited even in spatial tasks, and the processing philosophy is also different, as everything is centered on the event itself.

6.6.2 Advantages of Event-based vision

In chapter 2 we have shown how to adapt frame-based algorithm to deal with event-based data. The task is important as it allow us to reuse a large class of existing computer vision algorithms. We also proposed that the event-based data representation is a more general representation than the frame-based one. This is in part confirmed by the fact that classical computer vision can be performed on event-based data. When processing static images, such as pictures, the temporal data disappears, but when dealing with video stream, the event-based data is likely more powerful that the frame-based approach. This is already visible in chapter 4 where we perform stereo vision by exploiting the temporal dynamics of the signal. The possibility to study a visual scene with the additional temporal information about its dynamics is obviously improving many discrimination tasks. Object recognition is a very difficult task in classical frame-based computer vision. Humans are able to recognize objects in 2D pictures, but they likely learn the objects invariant key features through their past interactions with these objects. Recognizing a picture of an object we have never experienced is not so trivial. The extraction of these key features may be improved by the addition of the temporal dynamics in the model. Another important feature illustrated by this example relates to the interaction between different occurrences of the same object. Coding it as a stream of events might be more efficient for further recall and comparison than an equivalent frame-based code. Learning and recognition of real world objects is likely to involve more than one sensory modality. The event-based data representation is particularly suited for such sensory fusion. All modalities can indeed be encoded as events, sensory as well as motor.

6.6.3 Parallelization

The key principles for exploiting event-based data are related to parallelization. The computation is split by locally processing the events. Event-based computation is thus scalable and should run on parallel computers. This is very interesting as, in 2010, the current concern in processor development are all focused

on increasing the number of cores, and simplifying them. To preserve the Moor's law on the increase of performances in computers, there is a need for developing scalable algorithms able to exploit this ever increasing numbers of possible threads. The main problem in parallelizing algorithms resides in the sharing of the memory. Accessing shared variables is slowing down the computation but can also generates deadlocks. The event-based computation is thus not only a convenient way to parallelize the computation, but also a powerful unconventional computational framework. It is unconventional in the way memory is represented. The state of the computation is not represented as shared variables but as output events. In event-based computation, the local processing of incoming events results in the generation of an output event. We can easily shape the processing of the events as a network in which each computational unit is connected to others depending on the computation to perform. In this thesis, the network is a feed-forward sequence of filters, but in general it can be of any topology. The meaning, the state of the computation, is thus represented by both the connectivity of the network and the dynamics of the event stream. Events are inspired from neuronal spikes, and neuronal networks are particularly suited for running event-based computation.

6.6.4 Neuromorphic

In the same trend, but in advance in term of both numbers and simplification, the neuromorphic engineering field is actively working on producing neuronal hardware. The progress in neuromorphic design will not only lead to improved sensory devices such as the Dynamic Vision Sensors used in this thesis, but also to powerful neuronal network processors. Event-based algorithms are a key framework for the exploitation of such processors. These have applications in two important fields : electronic engineering and brain research. Implementing models of spike computation and confronting them with generated spike-like data is an important complement to the approach based on deriving models from biological data.

6.6.5 Mathematical framework

This is why it is crucial to develop a robust mathematical framework for event-based computation. This thesis is a first step in this direction, focusing on some aspects of vision but not all. We started by adapting classical computer vision so as to reuse and adapt as much as possible the existing framework. Future research efforts should follow this trend to produce a full event-based vision framework, extensible to any event-based computation.

6.6.6 Stereo vision

Our setup for stereo vision is very interesting to develop further. Its actual limitations are due to the variability in event generation between the two cameras. But with the recent progress in neuromorphic engineering, new cameras models will be providing higher spatial resolution and more information, such as measured grey levels, that can be used to add more robust spatial filters and increase the tolerance to missing or shifted events, while retaining the advantages of event-based computation. On-going work is focused on exploiting this event-based framework to develop automatic adaptive calibration of the setup. We can already dream of exploiting other features of a stereo setup such as variable focus and vergence to build a dynamic 3D scene reconstruction.

6.6.7 Tracking for monitoring

The application designed for behavioural monitoring is also promising despite current limitations in the accuracy, especially concerning the digits, due to poor spatial resolution mainly. The current version is nevertheless very easy to use and setup. The implicit compression of event-based data as well as the high temporal resolution of the DVS makes it the most experimentalist-friendly setup for high speed tracking. We have to polish the automatic tracking and extraction of trajectories for it to be used without any specialized technicians.

6.6.8 Difficulties, limitations

The work developed in this thesis was at the cross-over of many disciplines, the main ones being computer vision, neuromorphic engineering, computational neuroscience and projective geometry. It was a difficult task for many reasons, amongst which the need to address unconventional event-based computation without being hampered by previous knowledge on frame-based computing. We also tried many different approaches not documented in this thesis as they did not bring satisfying results but nevertheless took time and effort. The help and experience from many experts and students was really crucial to the success of this thesis. This work applies especially to the use of the Dynamic Vision Sensors, but should be of use for anyone who attempts event-based computation especially in the field of vision. The possibilities and applications than can be derived from the use of event-based vision are just starting to reveal themselves through the work and research done in the neuromorphic and event-based vision field.

A

Appendix

A.1 Files locations

The java classes implemented for this thesis are stored in the jAER repository at http://sourceforge.net/projects/jaer/. The java files are found in subfolders of /host/java/src while the matlab files are in /host/matlab/ratmonitoring.

A.1.1 Event-Based Algorithms

The event-based low pass filter is:
net.sf.jaer.eventprocessing.filter.LowPassFilterEventGenerator.java. The event-based thinning filter is:
net.sf.jaer.eventprocessing.filter.ThinningEventBased.java.

A.1.2 Event-based Stereo Vision: Calibration

The matlab calibration toolbox is stored in /host/matlab/ratmonitoring/calibration. The stereo display used to capture frame of the calibration pattern is:
ch.unizh.ini.jaer.projects.stereo3D.stereoDisplay.java. The epipolar rectification classes is:
net.sf.jaer.eventprocessing.filter.EpipolarRectification2.java.

A.1.3 Event-Based Stereo Vision: Correspondence Solving

The java file for disparity matching is is:
ch.unizh.ini.jaer.projects.stereo3D.StereoOnFundamentalMatrix4.java. The java file for 3D tracking is is:
ch.unizh.ini.jaer.projects.stereo3D.Binocular3DTracker.java.

A.1.4 Event-Based Monitoring of the Pellet Reaching Task

The matlab scripts to process the rat trajectories are stored in /host/matlab/rat-monitoring/pawtracking. The java file for the paw tracking is: ch.unizh.ini.jaer.projects.pawtracker.PawTrackerStereoBoard6.java.

List of Figures

List of Tables